

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

APPLICATION FOR LETTERS PATENT

**Application Program Interface for Network Software  
Platform**

Inventor(s):  
Brad Abrams

ATTORNEY'S DOCKET NO. MS1-865US

## **TECHNICAL FIELD**

This invention relates to network software, such as Web applications, and to computer software development of such network software. More particularly, this invention relates to an application program interface (API) that facilitates use of a network software platform by application programs and computer hardware.

## **BACKGROUND**

Very early on, computer software came to be categorized as “operating system” software or “application” software. Broadly speaking, an application is software meant to perform a specific task for the computer user such as solving a mathematical equation or supporting word processing. The operating system is the software that manages and controls the computer hardware. The goal of the operating system is to make the computer resources available to the application programmer while at the same time, hiding the complexity necessary to actually control the hardware.

The operating system makes the resources available via functions that are collectively known as the Application Program Interface or API. The term API is also used in reference to a single one of these functions. The functions are often grouped in terms of what resource or service they provide to the application programmer. Application software requests resources by calling individual API functions. API functions also serve as the means by which messages and information provided by the operating system are relayed back to the application software.

In addition to changes in hardware, another factor driving the evolution of operating system software has been the desire to simplify and speed application



1 software development. Application software development can be a daunting task,  
2 sometimes requiring years of developer time to create a sophisticated program  
3 with millions of lines of code. For a popular operating system such as Microsoft  
4 Windows®, application software developers write thousands of different  
5 applications each year that utilize the operating system. A coherent and usable  
6 operating system base is required to support so many diverse application  
7 developers.

8 Often, development of application software can be made simpler by making  
9 the operating system more complex. That is, if a function may be useful to several  
10 different application programs, it may be better to write it once for inclusion in the  
11 operating system, than requiring dozens of software developers to write it dozens  
12 of times for inclusion in dozens of different applications. In this manner, if the  
13 operating system supports a wide range of common functionality required by a  
14 number of applications, significant savings in applications software development  
15 costs and time can be achieved.

16 Regardless of where the line between operating system and application  
17 software is drawn, it is clear that for a useful operating system, the API between  
18 the operating system and the computer hardware and application software is as  
19 important as efficient internal operation of the operating system itself.

20 Over the past few years, the universal adoption of the Internet, and  
21 networking technology in general, has changed the landscape for computer  
22 software developers. Traditionally, software developers focused on single-site  
23 software applications for standalone desktop computers, or LAN-based computers  
24 that were connected to a limited number of other computers via a local area  
25 network (LAN). Such software applications were typically referred to as “shrink

1 wrapped" products because the software was marketed and sold in a shrink-  
2 wrapped package. The applications utilized well-defined APIs to access the  
3 underlying operating system of the computer.

4 As the Internet evolved and gained widespread acceptance, the industry  
5 began to recognize the power of hosting applications at various sites on the World  
6 Wide Web (or simply the "Web"). In the networked world, clients from anywhere  
7 could submit requests to server-based applications hosted at diverse locations and  
8 receive responses back in fractions of a second. These Web applications, however,  
9 were typically developed using the same operating system platform that was  
10 originally developed for standalone computing machines or locally networked  
11 computers. Unfortunately, in some instances, these applications do not adequately  
12 transfer to the distributed computing regime. The underlying platform was simply  
13 not constructed with the idea of supporting limitless numbers of interconnected  
14 computers.

15 To accommodate the shift to the distributed computing environment being  
16 ushered in by the Internet, Microsoft Corporation is developing a network  
17 software platform known as the ".NET" platform (read as "Dot Net"). The  
18 platform allows developers to create Web services that will execute over the  
19 Internet. Such a dynamic shift requires a new ground-up design of an entirely new  
20 API.

21 In response to this challenge, the inventors developed a unique set of API  
22 functions for Microsoft's .NET™ platform.  
23  
24  
25

## **SUMMARY**

An application program interface (API) provides a set of functions that make available support for processing XML documents for application developers who build Web applications on a network platform, such as Microsoft Corporation's .NET™ platform.

## **BRIEF DESCRIPTION OF THE DRAWINGS**

The same numbers are used throughout the drawings to reference like features.

Fig. 1 illustrates a network architecture in which clients access Web services over the Internet using conventional protocols.

Fig. 2 is a block diagram of a software architecture for Microsoft's .NET™ platform, which includes an application program interface (API).

Fig. 3 is a block diagram of unique namespaces supported by the API, as well as function classes of the various API functions.

Fig. 4 is a block diagram of an exemplary computer that may execute all or part of the software architecture.

## **BRIEF DESCRIPTION OF ACCOMPANYING COMPACT DISC**

Accompanying this specification is a compact disc that stores a compiled HTML help file identifying the API (application program interface) for Microsoft's .NET™ network platform. The file is named "cpref.chm" and was created on June 8, 2001. It is 30.81 Mbytes in size. The file can be executed on a Windows®-based computing device (e.g., IBM-PC, or equivalent) that executes a Windows®-brand operating system (e.g., Windows® NT, Windows® 98,

Windows® 2000, etc.). The compiled HTML help file stored on the compact disk is hereby incorporated by reference.

Additionally, the APIs contained in the compiled HTML help file are also provided in approximately 100 separate text files named "NamespaceName.txt". The text files comply with the ASCII format.

The compact disc itself is a CD-ROM, and conforms to the ISO 9660 standard.

### **DETAILED DESCRIPTION**

This disclosure addresses an application program interface (API) for a network platform upon which developers can build Web applications and services. More particularly, an exemplary API is described for the .NET™ platform created by Microsoft Corporation. The .NET™ platform is a software platform for Web services and Web applications implemented in the distributed computing environment. It represents the next generation of Internet computing, using open communication standards to communicate among loosely coupled Web services that are collaborating to perform a particular task.

In the described implementation, the .NET™ platform utilizes XML (extensible markup language), an open standard for describing data. XML is managed by the World Wide Web Consortium (W3C). XML is used for defining data elements on a Web page and business-to-business documents. XML uses a similar tag structure as HTML; however, whereas HTML defines how elements are displayed, XML defines what those elements contain. HTML uses predefined tags, but XML allows tags to be defined by the developer of the page. Thus, virtually any data items can be identified, allowing Web pages to function like

1 database records. Through the use of XML and other open protocols, such as  
2 Simple Object Access Protocol (SOAP), the .NET™ platform allows integration of  
3 a wide range of services that can be tailored to the needs of the user. Although the  
4 embodiments described herein are described in conjunction with XML and other  
5 open standards, such are not required for the operation of the claimed invention.  
6 Other equally viable technologies will suffice to implement the inventions  
7 described herein.

#### 9 EXEMPLARY NETWORK ENVIRONMENT

10 Fig. 1 shows a network environment 100 in which a network platform, such  
11 as the .NET™ platform, may be implemented. The network environment 100  
12 includes representative Web services 102(1), ..., 102(N), which provide services  
13 that can be accessed over a network 104 (e.g., Internet). The Web services,  
14 referenced generally as number 102, are programmable application components  
15 that are reusable and interact programmatically over the network 104, typically  
16 through industry standard Web protocols, such as XML, SOAP, WAP (wireless  
17 application protocol), HTTP (hypertext transport protocol), and SMTP (simple  
18 mail transfer protocol) although other means of interacting with the Web services  
19 over the network may also be used, such as Remote Procedure Call (RPC) or  
20 object broker type technology. A Web service can be self-describing and is often  
21 defined in terms of formats and ordering of messages.

22 Web services 102 are accessible directly by other services (as represented  
23 by communication link 106) or a software application, such as Web application  
24 110 (as represented by communication links 112 and 114). Each Web service 102  
25 is illustrated as including one or more servers that execute software to handle

1 requests for particular services. Such services often maintain databases that store  
2 information to be served back to requesters. Web services may be configured to  
3 perform any one of a variety of different services. Examples of Web services  
4 include login verification, notification, database storage, stock quoting, location  
5 directories, mapping, music, electronic wallet, calendar/scheduler, telephone  
6 listings, news and information, games, ticketing, and so on. The Web services can  
7 be combined with each other and with other applications to build intelligent  
8 interactive experiences.

9 The network environment 100 also includes representative client devices  
10 120(1), 120(2), 120(3), 120(4), ..., 120(M) that utilize the Web services 102 (as  
11 represented by communication link 122) and/or the Web application 110 (as  
12 represented by communication links 124, 126, and 128). The clients may  
13 communicate with one another using standard protocols as well, as represented by  
14 an exemplary XML link 130 between clients 120(3) and 120(4).

15 The client devices, referenced generally as number 120, can be  
16 implemented many different ways. Examples of possible client implementations  
17 include, without limitation, portable computers, stationary computers, tablet PCs,  
18 televisions/set-top boxes, wireless communication devices, personal digital  
19 assistants, gaming consoles, printers, photocopiers, and other smart devices.

20 The Web application 110 is an application designed to run on the network  
21 platform and may utilize the Web services 102 when handling and servicing  
22 requests from clients 120. The Web application 110 is composed of one or more  
23 software applications 130 that run atop a programming framework 132, which are  
24 executing on one or more servers 134 or other computer systems. Note that a  
25 portion of Web application 110 may actually reside on one or more of clients 120.

1 Alternatively, Web application 110 may coordinate with other software on clients  
2 120 to actually accomplish its tasks.

3 The programming framework 132 is the structure that supports the  
4 applications and services developed by application developers. It permits multi-  
5 language development and seamless integration by supporting multiple languages.  
6 It supports open protocols, such as SOAP, and encapsulates the underlying  
7 operating system and object model services. The framework provides a robust and  
8 secure execution environment for the multiple programming languages and offers  
9 secure, integrated class libraries.

10 The framework 132 is a multi-tiered architecture that includes an  
11 application program interface (API) layer 142, a common language runtime (CLR)  
12 layer 144, and an operating system/services layer 146. This layered architecture  
13 allows updates and modifications to various layers without impacting other  
14 portions of the framework. A common language specification (CLS) 140 allows  
15 designers of various languages to write code that is able to access underlying  
16 library functionality. The specification 140 functions as a contract between  
17 language designers and library designers. By adhering to the CLS, libraries  
18 written in one language can be directly accessible to code modules written in other  
19 languages to achieve seamless integration between code modules written in one  
20 language and code modules written in another language.

21 The API layer 142 presents groups of functions that the applications 130  
22 can call to access the resources and services provided by layer 146. By exposing  
23 the API functions for a network platform, application developers can create Web  
24 applications for distributed computing systems that make full use of the network  
25 resources and other Web services, without needing to understand the complex

1 interworkings of how those network resources actually operate or are made  
2 available. Moreover, the Web applications can be written in any number of  
3 programming languages, and translated into an intermediate language supported  
4 by the common language runtime 144 and included as part of the common  
5 language specification 140. . In this way, the API layer 142 can provide methods  
6 for a wide and diverse variety of applications.

7 Additionally, the framework 132 can be configured to support API calls  
8 placed by remote applications executing remotely from the servers 134 that host  
9 the framework. Representative applications 148(1) and 148(2) residing on clients  
10 120(3) and 120(M), respectively, can use the API functions by making calls  
11 directly, or indirectly, to the API layer 142 over the network 104.

12 The framework may also be implemented at the clients. Client 120(3)  
13 represents the situation where a framework 150 is implemented at the client. This  
14 framework may be identical to server-based framework 132, or modified for client  
15 purposes. Alternatively, the client-based framework may be condensed in the  
16 event that the client is a limited or dedicated function device, such as a cellular  
17 phone, personal digital assistant, handheld computer, or other  
18 communication/computing device.

## 20 DEVELOPERS' PROGRAMMING FRAMEWORK

21 Fig. 2 shows the programming framework 132 in more detail. The  
22 common language specification (CLS) layer 140 supports applications written in a  
23 variety of languages 130(1), 130(2), 130(3), 130(4), ..., 130(K). Such application  
24 languages include Visual Basic, C++, C#, COBOL, Jscript, Perl, Eiffel, Python,  
25 and so on. The common language specification 140 specifies a subset of features



or rules about features that, if followed, allow the various languages to communicate. For example, some languages do not support a given type (e.g., an “int\*” type) that might otherwise be supported by the common language runtime 144. In this case, the common language specification 140 does not include the type. On the other hand, types that are supported by all or most languages (e.g., the “int[]” type) is included in common language specification 140 so library developers are free to use it and are assured that the languages can handle it. This ability to communicate results in seamless integration between code modules written in one language and code modules written in another language. Since different languages are particularly well suited to particular tasks, the seamless integration between languages allows a developer to select a particular language for a particular code module with the ability to use that code module with modules written in different languages. The common language runtime 144 allow seamless multi-language development, with cross language inheritance, and provide a robust and secure execution environment for the multiple programming languages. For more information on the common language specification 140 and the common language runtime 144, the reader is directed to co-pending applications entitled “Method and System for Compiling Multiple Languages”, filed 6/21/2000 (serial number 09/598,105) and “Unified Data Type System and Method” filed 7/10/2000 (serial number 09/613,289), which are incorporated by reference.

The framework 132 encapsulates the operating system 146(1) (e.g., Windows®-brand operating systems) and object model services 146(2) (e.g., Component Object Model (COM) or Distributed COM). The operating system 146(1) provides conventional functions, such as file management, notification, event handling, user interfaces (e.g., windowing, menus, dialogs, etc.), security,

authentication, verification, processes and threads, memory management, and so on. The object model services 146(2) provide interfacing with other objects to perform various tasks. Calls made to the API layer 142 are handed to the common language runtime layer 144 for local execution by the operating system 146(1) and/or object model services 146(2).

The API 142 groups API functions into multiple namespaces. Namespaces essentially define a collection of classes, interfaces, delegates, enumerations, and structures, which are collectively called “types”, that provide a specific set of related functionality. A class represents managed heap allocated data that has reference assignment semantics. A delegate is an object oriented function pointer. An enumeration is a special kind of value type that represents named constants. A structure represents static allocated data that has value assignment semantics. An interface defines a contract that other types can implement.

By using namespaces, a designer can organize a set of types into a hierarchical namespace. The designer is able to create multiple groups from the set of types, with each group containing at least one type that exposes logically related functionality. In the exemplary implementation, the API 142 is organized into four root namespaces: a first namespace 200 for Web applications, a second namespace 202 for client applications, a third namespace 204 for data and XML, and a fourth namespace 206 for base class libraries (BCLs). Each group can then be assigned a name. For instance, types in the Web applications namespace 200 are assigned the name “Web”, and types in the data and XML namespace 204 can be assigned names “Data” and “XML” respectively. The named groups can be organized under a single “global root” namespace for system level APIs, such as an overall System namespace. By selecting and prefixing a top level identifier, the

types in each group can be easily referenced by a hierarchical name that includes the selected top level identifier prefixed to the name of the group containing the type. For instance, types in the Web applications namespace 200 can be referenced using the hierarchical name "System.Web". In this way, the individual namespaces 200, 202, 204, and 206 become major branches off of the System namespace and can carry a designation where the individual namespaces are prefixed with a designator, such as a "System." prefix.

The Web applications namespace 200 pertains to Web based functionality, such as dynamically generated Web pages (e.g., Microsoft's Active Server Pages (ASP)). It supplies types that enable browser/server communication. The client applications namespace 202 pertains to drawing and client side UI functionality. It supplies types that enable drawing of two-dimensional (2D) and three-dimensional (3D) drawings, imaging, and printing, as well as the ability to construct window forms, menus, boxes, and so on.

The data and XML namespace 204 relates to connectivity to data sources and XML functionality. It supplies classes, interfaces, delegates, and enumerations that enable security, specify data types, and serialize objects into XML format documents or streams. The base class libraries (BCL) namespace 206 pertains to basic system and runtime functionality. It contains the fundamental types and base classes that define commonly-used value and reference data types, events and event handlers, interfaces, attributes, and processing exceptions.

In addition to the framework 132, programming tools 210 are provided to assist the developer in building Web services and/or applications. One example of

1 the programming tools 200 is Visual Studio™, a multi-language suite of  
2 programming tools offered by Microsoft Corporation.

#### 3 4 ROOT API NAMESPACES

5 Fig. 3 shows the API 142 and its four root namespaces in more detail. In  
6 one embodiment, the namespaces are identified according to a hierarchical naming  
7 convention in which strings of names are concatenated with periods. For instance,  
8 the Web applications namespace 200 is identified by the root name  
9 “System.Web”. Within the “System.Web” namespace is another namespace for  
10 Web services, identified as “System.Web.Services”, which further identifies  
11 another namespace for a description known as  
12 “System.Web.Services.Description”. With this naming convention in mind, the  
13 following provides a general overview of selected namespaces of the API 142,  
14 although other naming conventions could be used with equal effect.

15 The Web applications namespace 200 (“System.Web”) defines additional  
16 namespaces, including:

- 17  
18 • A services namespace 300 (“System.Web.Services”) containing  
19 classes that enable a developer to build and use Web services. The  
20 services namespace 300 defines additional namespaces, including a  
21 description namespace 302 (“System.Web.Services.Description”)  
22 containing classes that enable a developer to publicly describe a  
23 Web service via a service description language (such as WSDL, a  
24 specification available from the W3C), a discovery namespace 304  
25 (“System.Web.Services.Discovery”) containing classes that allow

Web service consumers to locate available Web Services on a Web server, and a protocols namespace 306 ("System.Web.Services.Protocols") containing classes that define the protocols used to transmit data across a network during communication between Web service clients and the Web service itself.

- A caching namespace 308 ("System.Web.Caching") containing classes that enable developers to decrease Web application response time through temporarily caching frequently used resources on the server. This includes ASP.NET pages, web services, and user controls. (ASP.NET is the updated version of Microsoft's ASP technology.) Additionally, a cache dictionary is available for developers to store frequently used resources, such as hash tables and other data structures.
- A configuration namespace 310 ("System.Web.Configuration") containing classes that are used to read configuration data in for an application.
- A UI namespace 312 ("System.Web.UI") containing types that allow developers to create controls and pages that will appear in Web applications as user interfaces on a Web page. This namespace includes the control class, which provides all web based controls, whether those encapsulating HTML elements, higher level Web controls, or even custom User controls, with a common set of functionality. Also provided are classes which provide the web forms server controls data binding functionality, the ability to save

the view state of a given control or page, as well as parsing functionality for both programmable and literal controls. Within the UI namespace 312 are two additional namespaces: an HTML controls namespace 314 (“System.Web.UI.HtmlControls”) containing classes that permit developers to interact with types that encapsulates html 3.2 elements create HTML controls, and a Web controls namespace 316 (“System.Web.UI.WebControls”) containing classes that allow developers to create higher level Web controls.

- A security namespace 318 (“System.Web.Security”) containing classes used to implement security in web server applications, such as basic authentication, challenge response authentication, and role based authentication.
- A session state namespace 320 (“System.Web.SessionState”) containing classes used to access session state values (i.e., data that lives across requests for the lifetime of the session) as well as session-level settings and lifetime management methods.

The client applications namespace 202 is composed of two namespaces:

- A windows forms namespace 322 (“System.Windows.Forms”) containing classes for creating Windows®-based client applications that take full advantage of the rich user interface features available in the Microsoft Windows® operating system, such as the ability to drag and drop screen elements. Such classes may include wrapped

APIs available in the Microsoft Windows® operating system that are used in a windowing UI environment. Within this namespace are a design namespace 324 (“System.Windows.Forms.Design”) that contains classes to extend design-time support for Windows forms and a component model namespace 326 (“System.Windows.Forms.ComponentModel”) that contains the windows form implementation of the general component model defined in System.ComponentModel. This namespace contains designer tools, such as Visual Studio, which offer a rich experience for developers at design time.

- A drawing namespace 328 (“System.Drawing”) containing classes for graphics functionality. The drawing namespace 328 includes a 2D drawing namespace 330 (“System.Drawing.Drawing2D”) that contains classes and enumerations to provide advanced 2-dimensional and vector graphics functionality, an imaging namespace 332 (“System.Drawing.Imaging”) that contains classes for advanced imaging functionality, a printing namespace 334 (“System.Drawing.Printing”) that contains classes to permit developers to customize printing, and a text namespace 336 (“System.Drawing.Text”) that contains classes for advanced typography functionality.

The data and XML namespace 204 is composed of two namespaces:

- A data namespace 340 (“System.Data”) containing classes that enable developers to build components that efficiently manage data from multiple data sources. It implements an architecture that, in a disconnected scenario (such as the Internet), provides tools to request, update, and reconcile data in multiple tier systems. The data namespace 340 includes a common namespace 342 that contains types shared by data providers. A data provider describes a collection of types used to access a data source, such as a database, in the managed space. The data namespace 340 also includes an OLE DB namespace 344 that contains types pertaining to data used in object-oriented databases (e.g., Microsoft’s SQL Server), and a SQL client namespace 346 that contains types pertaining to data used by SQL clients. The data namespace also includes a SQL types namespace 348 (“System.Data.SqlTypes”) that contains classes for native data types within Microsoft’s SQL Server. The classes provide a safer, faster alternative to other data types. Using the objects within this namespace helps prevent type conversion errors caused in situations where loss of precision could occur. Because other data types are converted to and from SQL types behind the scenes, explicitly creating and using objects within this namespace results in faster code as well.
- An XML namespace 350 (“System.XML”) containing classes that provide standards-based support for processing XML. The supported standards include XML (e.g., version 1.0), XML Namespaces (both stream level and DOM), XML Schemas, XPath expressions, XSL/T



transformations, DOM Level 2 Core, and SOAP (e.g., version 1.1). The XML namespace 350 includes an XSLT namespace 352 ("System.XML.Xsl") that contains classes and enumerations to support XSLT (Extensible Stylesheet Language Transformations), an Xpath namespace 354 ("System.XML.Xpath") that contains an XPath parser and evaluation engine, and a serialization namespace 356 ("System.XML.Serialization") that contains classes used to serialize objects into XML format documents or streams.

The base class library namespace 206 ("System") includes the following namespaces:

- A collections namespace 360 ("System.Collections") containing interfaces and classes that define various collections of objects, such as lists, queues, arrays, hash tables and dictionaries.
- A configuration namespace 362 ("System.Configuration") containing classes and interfaces that allow developers to programmatically access configuration settings and handle errors in configuration files.
- A diagnostics namespace 364 ("System.Diagnostics") containing classes that are used to debug applications and to trace code execution. The namespace allows developers to start system processes, read and write to event logs, and monitor system performance using performance counters.

- A globalization namespace 366 (“System.Globalization”) containing classes that define culture-related information, including the language, the country/region, the calendars in use, the format patterns for dates, currency and numbers, and the sort order for strings.
- An I/O namespace 368 (“System.IO”) containing the infrastructure pieces to operate with the input/output of data streams, files, and directories. This namespace includes a model for working with streams of bytes, higher level readers and writers which consume those bytes, various constructions or implementations of the streams (e.g., FileStream and MemoryStream) and, a set of utility classes for working with files and directories.
- A net namespace 370 (“System.Net”) providing an extensive set of classes for building network-enabled application, referred to as the Net Class Libraries (NCL). One element to the design of the Net Class Libraries is an extensible, layered approach to exposing networking functionality. The NCL stack contains three basic layers. A base layer (System.Net.Socket) provides access to an interface to TCP/IP, the communications protocol of UNIX networks and the Internet. One example of such an interface is the “WinSock API” from Microsoft Corporation. The next layer is the Transport Protocol classes, which support such transport protocols as TCP and UDP. Developers may write their own protocol classes to provide support for protocols such as IGMP and ICMP. The third layer is the Web request, which provides an abstract factory pattern for the

creation of other protocol classes. The NCL provides implementations for Hyper Text Transport Protocol (HTTP).

- A reflection namespace (“System.Reflection”) 372 containing types that provide a managed view of loaded types, methods, and fields, with the ability to dynamically create and invoke types.
- A resources namespace 374 (“System.Resources”) containing classes and interfaces that allow developers to create, store and manage various culture-specific resources used in an application.
- A security namespace 376 (“System.Security”) supporting the underlying structure of the security system, including interfaces, attributes, exceptions, and base classes for permissions.
- A service process namespace 378 (“System.ServiceProcess”) containing classes that allow developers to install and run services. Services are long-running executables that run without a user interface. They can be installed to run under a system account that enables them to be started at computer reboot. Services whose implementation is derived from processing in one class can define specific behavior for start, stop, pause, and continue commands, as well as behavior to take when the system shuts down.
- A text namespace 380 (“System.Text”) containing classes representing various types of encodings (e.g., ASCII, Unicode, UTF-7, and UTF-8), abstract base classes for converting blocks of characters to and from blocks of bytes, and a helper class that manipulates and formats string objects without creating intermediate instances.

- A threading namespace 382 (“System.Threading”) containing classes and interfaces that enable multi-threaded programming. The threading namespace includes a ThreadPool class that manages groups of threads, a Timer class that enables a delegate to be called after a specified amount of time, and a Mutex class for synchronizing mutually-exclusive threads. This namespace also provides classes for thread scheduling, wait notification, and deadlock resolution.
- A runtime namespace 384 (“System.Runtime”) containing multiple namespaces concerning runtime features, including an interoperation services namespace 386 (“System.Runtime.InteropServices”) that contains a collection of classes useful for accessing COM objects. The types in the InteropServices namespace fall into the following areas of functionality: attributes, exceptions, managed definitions of COM types, wrappers, type converters, and the Marshal class. The runtime namespace 384 further includes a remoting namespace 388 (“System.Runtime.Remoting”) that contains classes and interfaces allowing developers to create and configure distributed applications. Another namespace within the runtime namespace 384 is a serialization namespace 390 (“System.Runtime.Serialization”) that contains classes used for serializing and deserializing objects. Serialization is the process of converting an object or a graph of objects into a linear sequence of bytes for either storage or transmission to another location.

Portions of the XML namespace 350 ("System.XML") are discussed in additional detail below.

#### SYSTEM.XML NAMESPACE

This is the overall namespace for the XML classes that provide standards-based support for processing XML. Various functionality for processing XML documents can be invoked via the XML namespace, such as non-cached forward only access to XML data, read-only random access to a data store, transforming of XML data using an XSLT stylesheet, constructing and editing of schemas, resolving of external XML resources named by a Uniform Resource Identifier (URI), storage and retrieval (as well as manipulation) of structured data through a relational dataset, non-cached forward only generation of streams and files containing XML data, DTD and XDR (as well as XSD) schema validation, and so forth. In one exemplary implementation, the standards supported by the namespace are:

- XML 1.0 - <http://www.w3.org/TR/1998/REC-xml-19980210> - including DTD support (XmlTextReader);
- XML Namespaces - <http://www.w3.org/TR/REC-xml-names/> - both stream level and DOM;
- XML Schemas - <http://www.w3.org/TR/xmlschema-1/> - supported for schema mapping and serialization. (see also XmlSchemaCollection which provides XDR schema validation);
- XPath expressions - <http://www.w3.org/TR/xpath> (XmlNavigator);
- XSL/T transformations - <http://www.w3.org/TR/xslt> (XslTransform);

- DOM Level 2 Core - <http://www.w3.org/TR/DOM-Level-2/> - for (XmlDocument);
- SOAP 1.1 - <http://msdn.microsoft.com/xml/general/soapspec.asp> (including the Soap Contract Language and Soap Discovery) used in XML object serialization.

The System.Xml namespace also includes a System.Xml.Schema namespace that contains XML classes that provide standards-based support for XML schemas. In one exemplary implementation, the supported standards are:

- XML Schemas for Structures - <http://www.w3.org/TR/xmlschema-1/> - supports for schema mapping and for validation. (see also XmlSchemaCollection which provides XSD and XDR schema validation); and
- XML Schemas for Data Types - <http://www.w3.org/TR/xmlschema-2/> - supports data types for XML schema definitions. (see also XmlSchemaCollection which provides XSD and XDR schema validation).

The following is a more detailed description of the System.Xml namespace, identifying various classes, interfaces, enumerations, and so forth contained in the System.Xml namespace (as well as the System.Xml.Schema, System.Xml.Serialization, System.Xml.XPath, and System.Xml.Xsl namespaces contained therein).

### **System.Xml**

This is the overall namespace for the XML classes that provide standards-based support for processing XML. The supported standards are: XML 1.0 -

1 <http://www.w3.org/TR/1998/REC-xml-19980210> - including DTD support  
2 (XmlTextReader) XML Namespaces - <http://www.w3.org/TR/REC-xml-names/> -  
3 both stream level and DOM.

#### 4 5 *Description*

6 This is the overall namespace for the XML classes that provide standards-  
7 based support for processing XML. The supported standards are: XML 1.0 -  
8 <http://www.w3.org/TR/1998/REC-xml-19980210> - including DTD support  
9 (XmlTextReader)XML Namespaces - <http://www.w3.org/TR/REC-xml-names/> -  
10 both stream level and DOM.

11 XmlDataDocument class (System.Xml)

#### 12 13 14 *Description*

15 Allows structured data to be stored, retrieved, and manipulated through a  
16 relational **System.Data.DataSet** .

17 This class extends **System.Xml.XmlDocument** . It enables you to load  
18 either relational data or XML data and manipulate that data using the W3C  
19 Document Object Model (DOM). The DOM presents data as a hierarchy of node  
20 objects. Because **XmlDataDocument** implements the  
21 **System.Xml.XPath.IXPathNavigable** interface it can also be used as the source  
22 document for the **System.Xml.Xsl.XslTransform** class.

23 Constructors:

24 XmlDataDocument

25 *Example Syntax:*

```

1
2 [C#] public XmlDataDocument();
3 [C++] public: XmlDataDocument();
4 [VB] Public Sub New()
5 [JScript] public function XmlDataDocument(); Initializes a new instance of the
6 XmlDataDocument class.

```

### *Description*

Initializes a new instance of the **XmlDataDocument** class.

The first time the **System.Xml.XmlDataDocument.DataSet** property is used, a new **DataSet** is created and associated with the **XmlDataDocument**.

**XmlDataDocument**

### *Example Syntax:*

```

15 [C#] public XmlDataDocument(DataSet dataset);
16 [C++] public: XmlDataDocument(DataSet* dataset);
17 [VB] Public Sub New(ByVal dataset As DataSet)
18 [JScript] public function XmlDataDocument(dataset : DataSet);

```

### *Description*

Initializes a new instance of the **XmlDataDocument** class with the specified **System.Data.DataSet**.

The **XmlDataDocument** is synchronized with the specified **DataSet**. Any data in the **DataSet** is immediately available through the **XmlDataDocument**. Any changes in the **DataSet** are propagated in the **XmlDataDocument**. Any



changes made in the **XmlDataDocument** , provided they match the **DataSet** schema, are propagated in the **DataSet** . The **DataSet** to load into the **XmlDataDocument** .

Properties:

Attributes

BaseURI

ChildNodes

DataSet

### *Description*

Gets a **System.Data.DataSet** that provides a relational representation of the data in the **XmlDataDocument** .

The **DataSet** enables you to access the data in the **XmlDataDocument** using a relational model. This means that you can handle the data as tables and views, rows and columns, relations, and so on.

DocumentElement

DocumentType

FirstChild

HasChildNodes

Implementation

InnerText

InnerXml

IsReadOnly

Item

1	Item
2	LastChild
3	LocalName
4	Name
5	NamespaceURI
6	NameTable
7	NextSibling
8	NodeType
9	OuterXml
10	OwnerDocument
11	ParentNode
12	Prefix
13	PreserveWhitespace
14	PreviousSibling
15	Value
16	XmlResolver
17	Methods:
18	CloneNode
19	
20	[C#] public override XmlNode CloneNode(bool deep);
21	[C++] public: XmlNode* CloneNode(bool deep);
22	[VB] Overrides Public Function CloneNode(ByVal deep As Boolean) As
23	XmlNode
24	[JScript] public override function CloneNode(deep : Boolean) : XmlNode;
25	

1  
2 *Description*

3 Creates a duplicate of the current node.

4 *Return Value:* The cloned node.

5 Cloning the **XmlDataDocument** also clones the **System.Data.DataSet**  
6 schema. **true** to recursively clone the subtree under the specified node; **false** to  
7 clone only the node itself.

8 **CreateElement**

9  
10 [C#] public override XmlElement CreateElement(string prefix, string localName,  
11 string namespaceURI);

12 [C++] public: XmlElement\* CreateElement(String\* prefix, String\* localName,  
13 String\* namespaceURI);

14 [VB] Overrides Public Function CreateElement(ByVal prefix As String, ByVal  
15 localName As String, ByVal namespaceURI As String) As XmlElement

16 [JScript] public override function CreateElement(prefix : String, localName :  
17 String, namespaceURI : String) : XmlElement; Creates an  
18 **System.Xml.XmlElement** .

19  
20 *Description*

21 Creates an element with the specified **System.Xml.XmlNode.Prefix** ,  
22 **System.Xml.XmlDocument.LocalName** , and  
23 **System.Xml.XmlNode.NamespaceURI** .

24 *Return Value:* A new **System.Xml.XmlElement** .

**Notes to Inheritors** If you overload this function, it should not be used for extensibility. Instead, you should return an element created by the base class, **XmlDataDocument** in this case. See the following example. The prefix of the new element; if String.Empty or **null** there is no prefix. The local name of the new element. The namespace URI of the new element; if String.Empty or **null** there is no namespaceURI.

#### CreateEntityReference

```
[C#] public override XmlEntityReference CreateEntityReference(string name);
[C++] public: XmlEntityReference* CreateEntityReference(String* name);
[VB] Overrides Public Function CreateEntityReference(ByVal name As String)
As XmlEntityReference
[JScript] public override function CreateEntityReference(name : String) :
XmlEntityReference;
```

#### *Description*

Creates an **System.Xml.XmlEntityReference** with the specified name.  
Name of the entity reference.

#### CreateNavigator

```
[C#] protected override XPathNavigator CreateNavigator(XmlNode node);
[C++] protected: XPathNavigator* CreateNavigator(XmlNode* node);
[VB] Overrides Protected Function CreateNavigator(ByVal node As XmlNode)
As XPathNavigator
[JScript] protected override function CreateNavigator(node : XmlNode) :
```

XPathNavigator; Creates a new **System.Xml.XPath.XPathNavigator** object for navigating this document.

#### *Description*

Creates a new **System.Xml.XPath.XPathNavigator** object for navigating this document.

*Return Value:* An **XPathNavigator** .

The **XPathNavigator** provides read-only, random access to data. Because it is optimized for XSLT transformations, it provides performance benefits when used as an input mechanism to the

**System.Xml.Xsl.XslTransform.Transform(System.Xml.XPath.IXPathNavigable, System.Xml.Xsl.XsltArgumentList)** method. The **System.Xml.XmlNode** you want to navigate.

#### *GetElementById*

[C#] public override XmlElement GetElementById(string elemId);

[C++] public: XmlElement\* GetElementById(String\* elemId);

[VB] Overrides Public Function GetElementById(ByVal elemId As String) As XmlElement

[JScript] public override function GetElementById(elemId : String) : XmlElement;

#### *Description*

Gets the **System.Xml.XmlElement** with the specified ID. The attribute ID to match.

#### *GetElementFromRow*

```

[C#] public XmlElement GetElementFromRow(DataRow r);
[C++] public: XmlElement* GetElementFromRow(DataRow* r);
[VB] Public Function GetElementFromRow(ByVal r As DataRow) As
XmlElement
[JScript] public function GetElementFromRow(r : DataRow) : XmlElement;
    
```

#### *Description*

Retrieves the **System.Xml.XmlElement** associated with the specified **System.Data.DataRow** .

*Return Value:* The **XmlElement** containing a representation of the specified **DataRow** . The **DataRow** whose associated **XmlElement** you wish to retrieve.

#### *GetRowFromElement*

```

[C#] public DataRow GetRowFromElement(XmlElement e);
[C++] public: DataRow* GetRowFromElement(XmlElement* e);
[VB] Public Function GetRowFromElement(ByVal e As XmlElement) As
DataRow
[JScript] public function GetRowFromElement(e : XmlElement) : DataRow;
    
```

#### *Description*

Retrieves the **System.Data.DataRow** associated with the specified **System.Xml.XmlElement** .

*Return Value:* The **DataRow** containing a representation of the **XmlElement** . The **XmlElement** whose associated **DataRow** you wish to retrieve.

## Load

[C#] public override void Load(XmlReader reader);  
[C++] public: void Load(XmlReader\* reader);  
[VB] Overrides Public Sub Load(ByVal reader As XmlReader)  
[JScript] public override function Load(reader : XmlReader); Loads the **XmlDataDocument** using the specified data source and synchronizes the **System.Data.DataSet** with the loaded data.

### *Description*

Loads the **XmlDataDocument** from the specified **System.Xml.XmlReader** .  
**System.Xml.XmlWhitespace** nodes are not created unless **System.Xml.XmlDocument.PreserveWhitespace** is set to **true** . Also, if the reader has been configured to not return whitespace, then no whitespace nodes will be created. In other words **Load** does not change the whitespace handling of the given reader. **XmlReader** containing the XML document to load.

## WriteTo

[C#] public override void WriteTo(XmlWriter writer);  
[C++] public: void WriteTo(XmlWriter\* writer);  
[VB] Overrides Public Sub WriteTo(ByVal writer As XmlWriter)  
[JScript] public override function WriteTo(writer : XmlWriter);

### *Description*

Saves the current node to the specified **System.Xml.XmlWriter** . The **XmlWriter** to which you want to save.

EntityHandling enumeration (System.Xml)

WriteTo

#### *Description*

Specifies how entities are handled.

WriteTo

[C#] public const EntityHandling ExpandCharEntities;

[C++] public: const EntityHandling ExpandCharEntities;

[VB] Public Const ExpandCharEntities As EntityHandling

[JScript] public var ExpandCharEntities : EntityHandling;

#### *Description*

Expands character entities and returns general entities as nodes (

**System.Xml.XmlValidatingReader.NodeType**

=XmlNodeType.EntityReference, **System.Xml.XmlValidatingReader.Name**

=the name of the entity, **System.Xml.XmlValidatingReader.HasValue** = false).

WriteTo

[C#] public const EntityHandling ExpandEntities;

[C++] public: const EntityHandling ExpandEntities;

[VB] Public Const ExpandEntities As EntityHandling



1 [JScript] public var ExpandEntities : EntityHandling;

2  
3 *Description*

4 Expands all entities. This is the default.

5 Formatting enumeration (System.Xml)

6 ToString

7  
8  
9 *Description*

10 Specifies formatting options for the **System.Xml.XmlTextWriter** .

11 ToString

12  
13 [C#] public const Formatting Indented;

14 [C++] public: const Formatting Indented;

15 [VB] Public Const Indented As Formatting

16 [JScript] public var Indented : Formatting;

17  
18 *Description*

19 Causes child elements to be indented according to the  
20 **System.Xml.XmlTextWriter.Indentation** and  
21 **System.Xml.XmlTextWriter.IndentChar** settings. This option indents element  
22 content only; mixed content is not affected. For the XML 1.0 definitions of these  
23 terms, see the W3C documentation ([http://www.w3.org/TR/1998/REC-xml-](http://www.w3.org/TR/1998/REC-xml-19980210#sec-element-content)  
24 [19980210#sec-element-content](http://www.w3.org/TR/1998/REC-xml-19980210#sec-element-content) and [http://www.w3.org/TR/1998/REC-xml-](http://www.w3.org/TR/1998/REC-xml-19980210#sec-mixed-content)  
25 [19980210#sec-mixed-content](http://www.w3.org/TR/1998/REC-xml-19980210#sec-mixed-content)).

ToString

[C#] public const Formatting None;

[C++] public: const Formatting None;

[VB] Public Const None As Formatting

[JScript] public var None : Formatting;

### *Description*

No special formatting is applied. This is the default.

IHasXmlNode interface (System.Xml)

ToString

### *Description*

Enables a class to return an **System.Xml.XmlNode** from the current context or position.

This interface is implemented by classes which operate over classes that have **System.Xml.XmlNode** nodes. For example, if an **System.Xml.XPath.XPathNavigator** is implemented over an **System.Xml.XmlDocument** you can use the **System.Xml.IHasXmlNode.GetNode** method to access the node representing the current position of the navigator.

GetNode

[C#] XmlNode GetNode();

```

1 [C++] XmlNode* GetNode();
2 [VB] Function GetNode() As XmlNode
3 [JScript] function GetNode() : XmlNode;

```

#### *Description*

Returns the **System.Xml.XmlNode** for the current position.

*Return Value:* The **XmlNode** for the current position.

The following C# code uses **GetNode** to access a node the **System.Xml.XPath.XPathNavigator** is currently positioned on.

IXmlLineInfo interface (System.Xml)

GetNode

#### *Description*

Provides an interface to enable a class to return line and position information.

LineNumber

GetNode

```

20 [C#] int LineNumber {get;}
21 [C++] int get_LineNumber();
22 [VB] ReadOnly Property LineNumber As Integer
23 [JScript] abstract function get LineNumber() : int;

```

#### *Description*

Gets the current line number.

This property is used primarily for error reporting, but can be called at any time. The starting value is 1. Combined with **System.Xml.IXmlLineInfo.LinePosition**, a value of 1,1 indicates the start of a document.

LinePosition

GetNode

[C#] int LinePosition {get;}

[C++] int get\_LinePosition();

[VB] ReadOnly Property LinePosition As Integer

[JScript] abstract function get LinePosition() : int;

#### *Description*

Gets the current line position.

This property is used primarily for error reporting, but can be called at any time. The starting value is 1. Combined with **System.Xml.IXmlLineInfo.LineNumber**, a value of 1,1 indicates the start of a document.

HasLineInfo

[C#] bool HasLineInfo();

[C++] bool HasLineInfo();

[VB] Function HasLineInfo() As Boolean

[JScript] function HasLineInfo() : Boolean;

1  
2 *Description*

3 Gets a value indicating whether the class can return line information.

4 *Return Value:* **true** if **System.Xml.IXmlLineInfo.LineNumber** and  
5 **System.Xml.IXmlLineInfo.LinePosition** can be provided; otherwise, **false** .

6 NameTable class (System.Xml)

7 HasLineInfo

8  
9  
10 *Description*

11 Implements a single-threaded **System.Xml.XmlNameTable** .

12 Several classes, such as **System.Xml.XmlDocument** and  
13 **System.Xml.XmlReader** use the **NameTable** class internally, to store attribute  
14 and element names. When an element or attribute name occurs multiple times in  
15 an XML document it is stored only once in the **NameTable** . This allows you to  
16 do object comparisons on these strings rather than a more expensive string  
17 comparison.

18 NameTable

19 *Example Syntax:*

20 HasLineInfo

21  
22 [C#] public NameTable();

23 [C++] public: NameTable();

24 [VB] Public Sub New()

25 [JScript] public function NameTable();

## Description

Initializes a new instance of the **NameTable** class.

Constructs an empty NameTable.

Add

[C#] public override string Add(string key);

[C++] public: String\* Add(String\* key);

[VB] Overrides Public Function Add(ByVal key As String) As String

[JScript] public override function Add(key : String) : String; Atomizes the specified string and adds it to the **NameTable** .

## Description

Atomizes the specified string and adds it to the **NameTable** .

*Return Value:* The atomized string or the existing string if it already exists in the **NameTable** . The string to add.

Add

[C#] public override string Add(char[] key, int start, int len);

[C++] public: String\* Add(\_\_wchar\_t key \_\_gc[], int start, int len);

[VB] Overrides Public Function Add(ByVal key() As Char, ByVal start As Integer, ByVal len As Integer) As String

[JScript] public override function Add(key : Char[], start : int, len : int) : String;

## Description

Atomizes the specified string and adds it to the **NameTable** .

*Return Value:* The atomized string or the existing atom if one already exists in the **NameTable** . If *len* is zero, String.Empty is returned. The character array containing the string to add. The zero-based index into the array specifying the first character of the string. The number of characters in the string.

Get

[C#] public override string Get(string value);

[C++] public: String\* Get(String\* value);

[VB] Overrides Public Function Get(ByVal value As String) As String

[JScript] public override function Get(value : String) : String; Gets the atomized string object.

*Description*

Gets the atomized string object with the specified value.

*Return Value:* The atomized string object or **null** if the string has not already been atomized. The name to find.

Get

[C#] public override string Get(char[] key, int start, int len);

[C++] public: String\* Get(\_\_wchar\_t key \_\_gc[], int start, int len);

[VB] Overrides Public Function Get(ByVal key() As Char, ByVal start As Integer, ByVal len As Integer) As String

[JScript] public override function Get(key : Char[], start : int, len : int) : String;

1  
2 *Description*

3 Gets the atomized **String** object containing the same characters as the  
4 specified range of characters in the given array.

5 *Return Value:* The atomized string or **null** if the string has not already been  
6 atomized. If *len* is zero, **String.Empty** is returned. The character array containing  
7 the name to find. The zero-based index into the array specifying the first character  
8 of the name. The number of characters in the name.

9 ReadState enumeration (System.Xml)

10 ToString

11  
12  
13 *Description*

14 Specifies the state of the reader.

15 ToString

16  
17 [C#] public const ReadState Closed;

18 [C++] public: const ReadState Closed;

19 [VB] Public Const Closed As ReadState

20 [JScript] public var Closed : ReadState;

21  
22 *Description*

23 The **System.Xml.XmlReader.Close** method has been called.

24 ToString



1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25

```
[C#] public const ReadState EndOfFile;
[C++] public: const ReadState EndOfFile;
[VB] Public Const EndOfFile As ReadState
[JScript] public var EndOfFile : ReadState;
```

*Description*

The end of the file has been reached successfully.

ToString

```
[C#] public const ReadState Error;
[C++] public: const ReadState Error;
[VB] Public Const Error As ReadState
[JScript] public var Error : ReadState;
```

*Description*

An error occurred that prevents the read operation from continuing.

ToString

```
[C#] public const ReadState Initial;
[C++] public: const ReadState Initial;
[VB] Public Const Initial As ReadState
[JScript] public var Initial : ReadState;
```

*Description*

The **Read** method has not been called.

ToString

[C#] public const ReadState Interactive;

[C++] public: const ReadState Interactive;

[VB] Public Const Interactive As ReadState

[JScript] public var Interactive : ReadState;

### *Description*

The **Read** method has been called. Additional methods may be called on the reader.

ValidationType enumeration (System.Xml)

ToString

### *Description*

Specifies the type of validation to perform.

The validation model has three characteristics, strict, informative, and status. Strict, does not allow the mixing of validation types, informative provides a warning if the schema or DTD cannot be found, and status provides warnings if validation cannot be performed for elements and attributes from schemas.

ToString

[C#] public const ValidationType Auto;

[C++] public: const ValidationType Auto;

[VB] Public Const Auto As ValidationType

[JScript] public var Auto : ValidationType;

#### *Description*

**System.Xml.XmlValidatingReader** validates if DTD or schema information is found.

ToString

[C#] public const ValidationType DTD;

[C++] public: const ValidationType DTD;

[VB] Public Const DTD As ValidationType

[JScript] public var DTD : ValidationType;

#### *Description*

Validates according to the DTD.

ToString

[C#] public const ValidationType None;

[C++] public: const ValidationType None;

[VB] Public Const None As ValidationType

[JScript] public var None : ValidationType;

#### *Description*

Creates an XML 1.0 compliant non-validating parser. Default attributes are reported and general entities can be resolved by calling

**System.Xml.XmlValidatingReader.ResolveEntity** . The DOCTYPE is not used for validation purposes.

ToString

[C#] public const ValidationType Schema;  
[C++] public: const ValidationType Schema;  
[VB] Public Const Schema As ValidationType  
[JScript] public var Schema : ValidationType;

#### *Description*

Validate according to XSD schemas; including inline schemas. XSD schemas are associated with namespace URIs either by using the **schemaLocation** attribute or the provided **System.Xml.XmlValidatingReader.Schemas** property.

ToString

[C#] public const ValidationType XDR;  
[C++] public: const ValidationType XDR;  
[VB] Public Const XDR As ValidationType  
[JScript] public var XDR : ValidationType;

#### *Description*

Validate according to XDR schemas; including inline schemas. XDR schemas are recognized using the **x-schema** namespace prefix or the **System.Xml.XmlValidatingReader.Schemas** property.

WhitespaceHandling enumeration (System.Xml)

ToString

*Description*

Specifies how whitespace is handled.

ToString

[C#] public const WhitespaceHandling All;

[C++] public: const WhitespaceHandling All;

[VB] Public Const All As WhitespaceHandling

[JScript] public var All : WhitespaceHandling;

*Description*

Return Whitespace and SignificantWhitespace nodes. This is the default.

ToString

[C#] public const WhitespaceHandling None;

[C++] public: const WhitespaceHandling None;

[VB] Public Const None As WhitespaceHandling

[JScript] public var None : WhitespaceHandling;

*Description*

Return no Whitespace and no SignificantWhitespace nodes.

ToString

[C#] public const WhitespaceHandling Significant;  
 [C++] public: const WhitespaceHandling Significant;  
 [VB] Public Const Significant As WhitespaceHandling  
 [JScript] public var Significant : WhitespaceHandling;

#### *Description*

Return Significant Whitespace nodes only.

WriteState enumeration (System.Xml)

ToString

#### *Description*

Specifies the state of the **System.Xml.XmlWriter** .

ToString

[C#] public const WriteState Attribute;  
 [C++] public: const WriteState Attribute;  
 [VB] Public Const Attribute As WriteState  
 [JScript] public var Attribute : WriteState;

#### *Description*

An attribute value is being written.

ToString

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25

```
[C#] public const WriteState Closed;  
[C++] public: const WriteState Closed;  
[VB] Public Const Closed As WriteState  
[JScript] public var Closed : WriteState;
```

*Description*

The **System.Xml.XmlWriter.Close** method has been called.

ToString

```
[C#] public const WriteState Content;  
[C++] public: const WriteState Content;  
[VB] Public Const Content As WriteState  
[JScript] public var Content : WriteState;
```

*Description*

The element content is being written.

ToString

```
[C#] public const WriteState Element;  
[C++] public: const WriteState Element;  
[VB] Public Const Element As WriteState  
[JScript] public var Element : WriteState;
```

*Description*

1 An element start tag is being written.

2 ToString

3

4 [C#] public const WriteState Prolog;

5 [C++] public: const WriteState Prolog;

6 [VB] Public Const Prolog As WriteState

7 [JScript] public var Prolog : WriteState;

8

9 *Description*

10 The prolog is being written.

11 ToString

12

13 [C#] public const WriteState Start;

14 [C++] public: const WriteState Start;

15 [VB] Public Const Start As WriteState

16 [JScript] public var Start : WriteState;

17

18 *Description*

19 A Write method has not been called.

20 XmlAttribute class (System.Xml)

21 ToString

22

23

24 *Description*

25



Represents an attribute. Valid and default values for the attribute are defined in a DTD or schema.

XmlAttribute

*Example Syntax:*

ToString

[C#] protected internal XmlAttribute(string prefix, string localName, string namespaceURI, XmlDocument doc);

[C++] internal: XmlAttribute(String\* prefix, String\* localName, String\* namespaceURI, XmlDocument\* doc);

[VB] Protected Friend Sub New(ByVal prefix As String, ByVal localName As String, ByVal namespaceURI As String, ByVal doc As XmlDocument)

[JScript] package function XmlAttribute(prefix : String, localName : String, namespaceURI : String, doc : XmlDocument);

#### *Description*

Attributes

BaseURI

ToString

#### *Description*

Gets the base URI of the node.

A networked XML document is comprised of chunks of data aggregated using various W3C standard inclusion mechanisms and therefore contains nodes

that come from different places. The **BaseURI** tells you where these nodes came from.

ChildNodes

FirstChild

HasChildNodes

InnerText

ToString

#### *Description*

Gets or sets the concatenated values of the node and all its children.

Setting this property replaces all the children with the parsed contents of the given string.

InnerXml

ToString

[C#] public override string InnerXml {get; set;}

[C++] public: \_\_property virtual String\* get\_InnerXml();public: \_\_property virtual void set\_InnerXml(String\*);

[VB] Overrides Public Property InnerXml As String

[JScript] public function get InnerXml() : String;public function set InnerXml(String);

#### *Description*

Gets or sets the markup representing just the children of this node.

This is also settable which replaces the children of the node with the parsed contents of the given string. The parsing is done in the current namespace context.

IsReadOnly

Item

Item

LastChild

LocalName

ToString

#### *Description*

Gets the local name of the node.

If the node does not have a prefix, **LocalName** is the same as

**System.Xml.XmlAttribute.Name** .

Name

ToString

[C#] public override string Name {get;}

[C++] public: \_\_property virtual String\* get\_Name();

[VB] Overrides Public ReadOnly Property Name As String

[JScript] public function get Name() : String;

#### *Description*

Gets the qualified name of the node.

NamespaceURI

1 ToString

2

3 [C#] public override string NamespaceURI {get;}

4 [C++] public: \_\_property virtual String\* get\_NamespaceURI();

5 [VB] Overrides Public ReadOnly Property NamespaceURI As String

6 [JScript] public function get NamespaceURI() : String;

7

8 *Description*

9 Gets the namespace URI of this node.

10 An attribute does not inherit its namespace from the element it is attached

11 to. If an attribute is not explicitly given a namespace, the namespace URI is

12 considered to be String.Empty.

13 NextSibling

14 NodeType

15 ToString

16

17

18 *Description*

19 Gets the type of the current node.

20 OuterXml

21 OwnerDocument

22 ToString

23

24

25 *Description*

Gets the **System.Xml.XmlDocument** to which this node belongs.

OwnerElement

ToString

[C#] public virtual XmlElement OwnerElement {get;}

[C++] public: \_\_property virtual XmlElement\* get\_OwnerElement();

[VB] Overridable Public ReadOnly Property OwnerElement As XmlElement

[JScript] public function get OwnerElement() : XmlElement;

#### *Description*

Gets the **System.Xml.XmlElement** that contains this attribute.

ParentNode

ToString

[C#] public override XmlNode ParentNode {get;}

[C++] public: \_\_property virtual XmlNode\* get\_ParentNode();

[VB] Overrides Public ReadOnly Property ParentNode As XmlNode

[JScript] public function get ParentNode() : XmlNode;

#### *Description*

Gets the parent of this node. For **XmlAttribute** nodes, this property always returns **null**.

Prefix

ToString

```

1
2 [C#] public override string Prefix {get; set;}
3 [C++] public: __property virtual String* get_Prefix();public: __property virtual
4 void set_Prefix(String*);
5 [VB] Overrides Public Property Prefix As String
6 [JScript] public function get Prefix() : String;public function set Prefix(String);
7

```

#### Description

Gets or sets the namespace prefix of this node.

Changing the prefix of an attribute that is known to have a default value does not create a new attribute with the default value and the original prefix.

PreviousSibling

Specified

ToString

#### Description

Gets a value indicating whether the attribute value was explicitly set.

The implementation is in charge of this property, not the user. If the user changes the value of the attribute (even if it ends up having the same value as the default/fixed value) then the specified flag is automatically flipped to **true** . To re-specify the attribute as the default/fixed value from the DTD, the user must delete the attribute. The implementation then makes a new attribute available with specified set to **false** and the default/fixed value (if one exists).

Value

ToString

[C#] public override string Value {get; set;}

[C++] public: \_\_property virtual String\* get\_Value();public: \_\_property virtual  
void set\_Value(String\*);

[VB] Overrides Public Property Value As String

[JScript] public function get Value() : String;public function set Value(String);

### *Description*

Gets or sets the value of the node.

CloneNode

[C#] public override XmlNode CloneNode(bool deep);

[C++] public: XmlNode\* CloneNode(bool deep);

[VB] Overrides Public Function CloneNode(ByVal deep As Boolean) As

XmlNode

[JScript] public override function CloneNode(deep : Boolean) : XmlNode;

### *Description*

Creates a duplicate of this node.

*Return Value:* The duplicate node.

This method serves as a copy constructor for nodes. The cloned node has no parent ( **System.Xml.XmlAttribute.ParentNode** returns **null** ). **true** to recursively clone the subtree under the specified node; **false** to clone only the node itself

## WriteContentTo

```
[C#] public override void WriteContentTo(XmlWriter w);  
[C++] public: void WriteContentTo(XmlWriter* w);  
[VB] Overrides Public Sub WriteContentTo(ByVal w As XmlWriter)  
[JScript] public override function WriteContentTo(w : XmlWriter);
```

### *Description*

Saves all the children of the node to the specified XmlWriter. The XmlWriter where you want to save the current node.

## WriteTo

```
[C#] public override void WriteTo(XmlWriter w);  
[C++] public: void WriteTo(XmlWriter* w);  
[VB] Overrides Public Sub WriteTo(ByVal w As XmlWriter)  
[JScript] public override function WriteTo(w : XmlWriter);
```

### *Description*

Saves the node to the specified XmlWriter. The XmlWriter where you want to save the node.

XmlAttributeCollection class (System.Xml)

## WriteTo

### *Description*



Represents a collection of attributes that can be accessed by name or index.

Count

ItemOf

WriteTo

### *Description*

Gets the attribute with the specified name.

This method is the same as calling

**System.Xml.XmlNamedNodeMap.GetNamedItem(System.String)** . The qualified name of the attribute.

ItemOf

WriteTo

[C#] public virtual XmlAttribute this[int i] {get;}

[C++] public: \_\_property virtual XmlAttribute\* get\_ItemOf(int i);

[VB] Overridable Public Default ReadOnly Property ItemOf(ByVal i As Integer)

As XmlAttribute

[JScript] returnValue = XmlAttributeCollectionObject.ItemOf(i); Gets the attribute with the specified name or index.

### *Description*

Gets the attribute with the specified index.

This method is the same as calling

**System.Xml.XmlNamedNodeMap.Item(System.Int32)** . The index of the attribute.

ItemOf

WriteTo

[C#] public virtual XmlAttribute this[string localName, string namespaceURI] {get;}

[C++] public: \_\_property virtual XmlAttribute\* get\_ItemOf(String\* localName, String\* namespaceURI);

[VB] Overridable Public Default ReadOnly Property ItemOf(ByVal localName As String, ByVal namespaceURI As String) As XmlAttribute

[JScript] returnValue = XmlAttributeCollectionObject.ItemOf(localName, namespaceURI);

#### *Description*

Gets the attribute with the specified local name and namespace URI.

This method is the same as calling

**System.Xml.XmlNamedNodeMap.GetNamedItem(System.String)** . The local name of the attribute. The namespace URI of the attribute.

#### Append

[C#] public virtual XmlAttribute Append(XmlAttribute node);

[C++] public: virtual XmlAttribute\* Append(XmlAttribute\* node);

[VB] Overridable Public Function Append(ByVal node As XmlAttribute) As

XmlAttribute

[JScript] public function Append(node : XmlAttribute) : XmlAttribute;

#### *Description*

Inserts the specified attribute as the last node in the collection.

If *node* is already in the collection, it is moved to the last position. If an attribute with the same name is already present in the collection, the original attribute is removed from the collection and *node* is added to the end of the collection. The **System.Xml.XmlAttribute** to insert.

#### *CopyTo*

[C#] public void CopyTo(XmlAttribute[] array, int index);

[C++] public: void CopyTo(XmlAttribute\* array[], int index);

[VB] Public Sub CopyTo(ByVal array() As XmlAttribute, ByVal index As Integer)

[JScript] public function CopyTo(array : XmlAttribute[], index : int);

#### *Description*

Copies all the **System.Xml.XmlAttribute** objects from this collection into the given array. The array that is the destination of the objects copied from this collection. The index in *array* where copying begins.

#### *InsertAfter*

[C#] public virtual XmlAttribute InsertAfter(XmlAttribute newNode, XmlAttribute refNode);

```

1  [C++] public: virtual XmlAttribute* InsertAfter(XmlAttribute* newNode,
2  XmlAttribute* refNode);
3  [VB] Overridable Public Function InsertAfter(ByVal newNode As XmlAttribute,
4  ByVal refNode As XmlAttribute) As XmlAttribute
5  [JScript] public function InsertAfter(newNode : XmlAttribute, refNode :
6  XmlAttribute) : XmlAttribute;

```

#### *Description*

Inserts the specified attribute immediately after the specified reference attribute.

If an attribute with the same name is already present in the collection, the original attribute is removed from the collection and *newNode* is inserted into the collection. The **System.Xml.XmlAttribute** to insert. The **System.Xml.XmlAttribute** that is the reference attribute.

#### *InsertBefore*

```

17 [C#] public virtual XmlAttribute InsertBefore(XmlAttribute newNode,
18 XmlAttribute refNode);
19 [C++] public: virtual XmlAttribute* InsertBefore(XmlAttribute* newNode,
20 XmlAttribute* refNode);
21 [VB] Overridable Public Function InsertBefore(ByVal newNode As XmlAttribute,
22 ByVal refNode As XmlAttribute) As XmlAttribute
23 [JScript] public function InsertBefore(newNode : XmlAttribute, refNode :
24 XmlAttribute) : XmlAttribute;

```

## Description

Inserts the specified attribute immediately before the specified reference attribute.

If an attribute with the same name is already present in the collection, the original attribute is removed from the collection and *newNode* is inserted into the collection. The **System.Xml.XmlAttribute** to insert. The **System.Xml.XmlAttribute** that is the reference attribute.

## Prepend

[C#] public virtual XmlAttribute Prepend(XmlAttribute node);

[C++] public: virtual XmlAttribute\* Prepend(XmlAttribute\* node);

[VB] Overridable Public Function Prepend(ByVal node As XmlAttribute) As XmlAttribute

[JScript] public function Prepend(node : XmlAttribute) : XmlAttribute;

## Description

Inserts the specified attribute as the first node in the collection.

If *node* is already in the collection, it is moved to the first position. If an attribute with the same name is already present in the collection, the original attribute is removed from the collection and *node* is added to the beginning of the collection. The **System.Xml.XmlAttribute** to insert.

## Remove

[C#] public virtual XmlAttribute Remove(XmlAttribute node);

1 [C++] public: virtual XmlAttribute\* Remove(XmlAttribute\* node);  
 2 [VB] Overridable Public Function Remove(ByVal node As XmlAttribute) As  
 3 XmlAttribute  
 4 [JScript] public function Remove(node : XmlAttribute) : XmlAttribute;

5  
 6 *Description*

7 Removes the specified attribute from the collection.

8 *Return Value:* The node removed or **null** if it is not found in the collection. The  
 9 **System.Xml.XmlAttribute** to remove.

10 RemoveAll

11  
 12 [C#] public virtual void RemoveAll();  
 13 [C++] public: virtual void RemoveAll();  
 14 [VB] Overridable Public Sub RemoveAll()  
 15 [JScript] public function RemoveAll();

16  
 17 *Description*

18 Removes all attributes from the collection.

19 RemoveAt

20  
 21 [C#] public virtual XmlAttribute RemoveAt(int i);  
 22 [C++] public: virtual XmlAttribute\* RemoveAt(int i);  
 23 [VB] Overridable Public Function RemoveAt(ByVal i As Integer) As  
 24 XmlAttribute  
 25 [JScript] public function RemoveAt(i : int) : XmlAttribute;

## Description

Removes the attribute corresponding to the specified index from the collection.

*Return Value:* Returns **null** if there is no attribute at the specified index. The index of the node to remove. The first node has index 0.

## SetNamedItem

[C#] public override XmlNode SetNamedItem(XmlNode node);

[C++] public: XmlNode\* SetNamedItem(XmlNode\* node);

[VB] Overrides Public Function SetNamedItem(ByVal node As XmlNode) As XmlNode

[JScript] public override function SetNamedItem(node : XmlNode) : XmlNode;

## Description

Adds a **System.Xml.XmlNode** using its **System.Xml.XmlNode.Name** property

*Return Value:* If the *node* replaces an existing node with the same name, the old node is returned; otherwise, **null** is returned. An attribute node to store in this collection . The node will later be accessible using the name of the node. If a node with that name is already present in the collection, it is replaced by the new one.

## ICollection.CopyTo

[C#] void ICollection.CopyTo(Array array, int index);

[C++] void ICollection::CopyTo(Array\* array, int index);

[VB] Sub CopyTo(ByVal array As Array, ByVal index As Integer) Implements  
ICollection.CopyTo

[JScript] function ICollection.CopyTo(array : Array, index : int);

XmlCDataSection class (System.Xml)

ToString

### *Description*

Represents a CDATA section.

CDATA sections are used to quote or escape blocks of text to keep that text  
from being interpreted as markup language.

XmlCDataSection

### *Example Syntax:*

ToString

[C#] protected internal XmlCDataSection(string data, XmlDocument doc);

[C++] internal: XmlCDataSection(String\* data, XmlDocument\* doc);

[VB] Protected Friend Sub New(ByVal data As String, ByVal doc As  
XmlDocument)

[JScript] package function XmlCDataSection(data : String, doc : XmlDocument);

### *Description*

Attributes

BaseURI

ChildNodes



1 Data  
 2 FirstChild  
 3 HasChildNodes  
 4 InnerText  
 5 InnerXml  
 6 IsReadOnly  
 7 Item  
 8 Item  
 9 LastChild  
 10 Length  
 11 LocalName  
 12 ToString

13  
 14  
 15 *Description*

16 Gets the local name of the node.

17 Name

18 ToString

19  
 20 [C#] public override string Name {get;}

21 [C++] public: \_\_property virtual String\* get\_Name();

22 [VB] Overrides Public ReadOnly Property Name As String

23 [JScript] public function get Name() : String;

24  
 25 *Description*

Gets the qualified name of the node.

NamespaceURI

NextSibling

NodeType

ToString

*Description*

Gets the type of the current node.

OuterXml

OwnerDocument

ParentNode

Prefix

PreviousSibling

Value

CloneNode

[C#] public override XmlNode CloneNode(bool deep);

[C++] public: XmlNode\* CloneNode(bool deep);

[VB] Overrides Public Function CloneNode(ByVal deep As Boolean) As

XmlNode

[JScript] public override function CloneNode(deep : Boolean) : XmlNode;

*Description*

Creates a duplicate of this node.

*Return Value:* The cloned node.

**CloneNode** serves as a copy constructor for nodes. For CDATA nodes, the cloned node always includes the data content. To see how this method behaves with other node types, see **System.Xml.XmlNode.CloneNode(System.Boolean)** method in the **XmlNode** class. **true** to recursively clone the subtree under the specified node; **false** to clone only the node itself.

#### WriteContentTo

[C#] public override void WriteContentTo(XmlWriter w);

[C++] public: void WriteContentTo(XmlWriter\* w);

[VB] Overrides Public Sub WriteContentTo(ByVal w As XmlWriter)

[JScript] public override function WriteContentTo(w : XmlWriter);

#### Description

Saves the node to the specified **System.Xml.XmlWriter**. The **XmlWriter** to which you want to save.

#### WriteTo

[C#] public override void WriteTo(XmlWriter w);

[C++] public: void WriteTo(XmlWriter\* w);

[VB] Overrides Public Sub WriteTo(ByVal w As XmlWriter)

[JScript] public override function WriteTo(w : XmlWriter);

#### Description

Saves the node to the specified **System.Xml.XmlWriter** . The **XmlWriter** to which you want to save.

**XmlCharacterData** class (System.Xml)

**WriteTo**

### *Description*

Provides text-manipulation methods that are used by several classes. This class is **abstract** .

**XmlCharacterData**

*Example Syntax:*

**WriteTo**

[C#] protected internal **XmlCharacterData**(string data, **XmlDocument** doc);

[C++] internal: **XmlCharacterData**(String\* data, **XmlDocument**\* doc);

[VB] Protected Friend Sub New(ByVal data As String, ByVal doc As **XmlDocument**)

[JScript] package function **XmlCharacterData**(data : String, doc : **XmlDocument**);

### *Description*

Attributes

BaseURI

ChildNodes

Data

**WriteTo**

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25

*Description*

Contains this node's data.

FirstChild

HasChildNodes

InnerText

WriteTo

*Description*

Gets or sets the concatenated values of the node and all its children.

InnerXml

IsReadOnly

Item

Item

LastChild

Length

WriteTo

*Description*

Gets the length of the data, in characters.

LocalName

Name

1	NamespaceURI
2	NextSibling
3	NodeType
4	OuterXml
5	OwnerDocument
6	ParentNode
7	Prefix
8	PreviousSibling
9	Value
10	WriteTo

11

12

13 *Description*

14 Gets or sets the value of the node.

15 AppendData

16

17 [C#] public virtual void AppendData(string strData);

18 [C++] public: virtual void AppendData(String\* strData);

19 [VB] Overridable Public Sub AppendData(ByVal strData As String)

20 [JScript] public function AppendData(strData : String);

21

22 *Description*

23 Appends the specified string to the end of the character data of the node.

24

25

When overriding **AppendData** in a derived class, in order for events to be fired correctly, you must call the base class's **AppendData** method. String data that is to be inserted into the existing string.

#### DecideXPNodeTypeForTextNodes

[C#] protected internal bool DecideXPNodeTypeForTextNodes(XmlNode node, ref XPathNodeType xnt);

[C++] protected public: bool DecideXPNodeTypeForTextNodes(XmlNode\* node, XPathNodeType\* xnt);

[VB] Protected Friend Dim Function DecideXPNodeTypeForTextNodes(ByVal node As XmlNode, ByRef xnt As XPathNodeType) As Boolean

[JScript] package function DecideXPNodeTypeForTextNodes(node : XmlNode, xnt : XPathNodeType) : Boolean;

#### *Description*

#### DeleteData

[C#] public virtual void DeleteData(int offset, int count);

[C++] public: virtual void DeleteData(int offset, int count);

[VB] Overridable Public Sub DeleteData(ByVal offset As Integer, ByVal count As Integer)

[JScript] public function DeleteData(offset : int, count : int);

#### *Description*

Remove a range of characters from the node.

When overriding **DeleteData** in a derived class, in order for events to be fired correctly, you must call the base class's **DeleteData** method. Offset, in characters, at which to start deleting string data. Number of characters to delete.

#### InsertData

[C#] public virtual void InsertData(int offset, string strData);

[C++] public: virtual void InsertData(int offset, String\* strData);

[VB] Overridable Public Sub InsertData(ByVal offset As Integer, ByVal strData As String)

[JScript] public function InsertData(offset : int, strData : String);

#### Description

Insert the specified string at the specified character offset.

When overriding **InsertData** in a derived class, in order for events to be fired correctly, you must call the base class's **InsertData** method. Offset, in characters, at which to insert the supplied string data. String data that is to be inserted into the existing string.

#### ReplaceData

[C#] public virtual void ReplaceData(int offset, int count, string strData);

[C++] public: virtual void ReplaceData(int offset, int count, String\* strData);

[VB] Overridable Public Sub ReplaceData(ByVal offset As Integer, ByVal count As Integer, ByVal strData As String)

[JScript] public function ReplaceData(offset : int, count : int, strData : String);



## Description

Replace the specified number of characters starting at the specified offset with the specified string.

When overriding **ReplaceData** in a derived class, in order for events to be fired correctly, you must call the base class's **ReplaceData** method. Offset, in characters, at which to start replacing string data. Number of characters to replace. New data that replaces the old string data.

## Substring

[C#] public virtual string Substring(int offset, int count);

[C++] public: virtual String\* Substring(int offset, int count);

[VB] Overridable Public Function Substring(ByVal offset As Integer, ByVal count As Integer) As String

[JScript] public function Substring(offset : int, count : int) : String;

## Description

Retrieves a substring of the full string from the specified range. Offset, in characters, from the beginning of the string. An offset of zero indicates copying from the start of the data. Number of characters to retrieve from the specified offset.

XmlComment class (System.Xml)

WriteTo

*Description*

Represents the content of an XML comment.

XmlComment

*Example Syntax:*

WriteTo

[C#] protected internal XmlComment(string comment, XmlDocument doc);

[C++] internal: XmlComment(String\* comment, XmlDocument\* doc);

[VB] Protected Friend Sub New(ByVal comment As String, ByVal doc As XmlDocument)

[JScript] package function XmlComment(comment : String, doc : XmlDocument);

*Description*

Attributes

BaseURI

ChildNodes

Data

FirstChild

HasChildNodes

InnerText

InnerXml

IsReadOnly

Item

1 Item  
2 LastChild  
3 Length  
4 LocalName  
5 WriteTo  
6  
7

8 *Description*

9 Gets the local name of the node.

10 Name

11 WriteTo  
12

13 [C#] public override string Name {get;}

14 [C++] public: \_\_property virtual String\* get\_Name();

15 [VB] Overrides Public ReadOnly Property Name As String

16 [JScript] public function get Name() : String;  
17

18 *Description*

19 Gets the qualified name of the node.

20 NamespaceURI

21 NextSibling

22 NodeType

23 WriteTo  
24  
25

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25

*Description*

Gets the type of the current node.

OuterXml

OwnerDocument

ParentNode

Prefix

PreviousSibling

Value

CloneNode

[C#] public override XmlNode CloneNode(bool deep);

[C++] public: XmlNode\* CloneNode(bool deep);

[VB] Overrides Public Function CloneNode(ByVal deep As Boolean) As

XmlNode

[JScript] public override function CloneNode(deep : Boolean) : XmlNode;

*Description*

Creates a duplicate of this node.

*Return Value:* The cloned node.

**CloneNode** serves as a copy constructor for nodes. For comment nodes, the cloned node always includes the text content. To see how this method behaves with other node types, see **System.Xml.XmlNode.CloneNode(System.Boolean)**

method in the **XmlNode** class. **true** to recursively clone the subtree under the specified node; **false** to clone only the node itself.

#### WriteContentTo

[C#] public override void WriteContentTo(XmlWriter w);

[C++] public: void WriteContentTo(XmlWriter\* w);

[VB] Overrides Public Sub WriteContentTo(ByVal w As XmlWriter)

[JScript] public override function WriteContentTo(w : XmlWriter);

#### *Description*

Saves all the children of the node to the specified **System.Xml.XmlWriter**. The **XmlWriter** to which you want to save.

#### WriteTo

[C#] public override void WriteTo(XmlWriter w);

[C++] public: void WriteTo(XmlWriter\* w);

[VB] Overrides Public Sub WriteTo(ByVal w As XmlWriter)

[JScript] public override function WriteTo(w : XmlWriter);

#### *Description*

Saves the node to the specified **System.Xml.XmlWriter**. The **XmlWriter** to which you want to save.

XmlConvert class (System.Xml)

#### WriteTo

1  
2  
3 *Description*

4 Encodes and decodes XML names and provides methods for converting  
5 between CLR types and XSD types.

6 XmlConvert

7 *Example Syntax:*

8 WriteTo

9  
10 [C#] public XmlConvert();

11 [C++] public: XmlConvert();

12 [VB] Public Sub New()

13 [JScript] public function XmlConvert();

14 DecodeName

15  
16 [C#] public static string DecodeName(string name);

17 [C++] public: static String\* DecodeName(String\* name);

18 [VB] Public Shared Function DecodeName(ByVal name As String) As String

19 [JScript] public static function DecodeName(name : String) : String;

20  
21 *Description*

22 Transforms an XML name into an ADO.NET object name (such as  
23 DataTable or DataColumn).

Decoding: When the method transforms an XML name into an ADO.NET object name, the decoding step follows the following rules: Names are decoded from left to right. The name to be transformed.

#### EncodeLocalName

[C#] public static string EncodeLocalName(string name);

[C++] public: static String\* EncodeLocalName(String\* name);

[VB] Public Shared Function EncodeLocalName(ByVal name As String) As String

[JScript] public static function EncodeLocalName(name : String) : String;

#### *Description*

Converts names, such as DataTable or DataColumn names, that contain characters that are not permitted in XML names to valid names.

*Return Value:* The encoded name.

This method is the same as

**System.Xml.XmlConvert.EncodeName(System.String)** except it also encodes the colon character, guaranteeing that this can be used as the LocalName part of a namespace qualified name. The name to be encoded.

#### EncodeName

[C#] public static string EncodeName(string name);

[C++] public: static String\* EncodeName(String\* name);

[VB] Public Shared Function EncodeName(ByVal name As String) As String

[JScript] public static function EncodeName(name : String) : String;

## Description

Converts names, such as DataTable or DataColumn names, that contain characters that are not permitted in XML names to valid names.

*Return Value:* Returns the name with any invalid characters replaced by an escape string.

This method translates invalid characters, such as spaces or half-width Katakana, that need to be mapped to XML names without the support or presence of schemas. The invalid characters are translated into escaped numeric entity encodings. A name to be translated.

### EncodeNmToken

[C#] public static string EncodeNmToken(string name);

[C++] public: static String\* EncodeNmToken(String\* name);

[VB] Public Shared Function EncodeNmToken(ByVal name As String) As String

[JScript] public static function EncodeNmToken(name : String) : String;

## Description

Verifies the name is valid according to the XML spec.

*Return Value:* The encoded name.

This method guarantees the name is valid according to the XML spec. For example, if you passed this method the invalid name "70+", it would return "a\_x003a\_b" which is a valid XML name. The name to be encoded.

### ToBoolean



```

1
2 [C#] public static bool ToBoolean(string s);
3 [C++] public: static bool ToBoolean(String* s);
4 [VB] Public Shared Function ToBoolean(ByVal s As String) As Boolean
5 [JScript] public static function ToBoolean(s : String) : Boolean;
6

```

### *Description*

Converts the **System.String** to a **System.Boolean** equivalent.

*Return Value:* A **Boolean** value (i.e **true** or **false** ).

Valid strings are "1" or "true" for **true** and "0" or "false" for **false** . The string to convert.

### **ToByte**

```

13
14 [C#] public static byte ToByte(string s);
15 [C++] public: static unsigned char ToByte(String* s);
16 [VB] Public Shared Function ToByte(ByVal s As String) As Byte
17 [JScript] public static function ToByte(s : String) : Byte;
18

```

### *Description*

Converts the **System.String** to a **System.Byte** equivalent.

*Return Value:* A **Byte** equivalent of the string. The string to convert.

### **ToChar**

```

23
24 [C#] public static char ToChar(string s);
25 [C++] public: static __wchar_t ToChar(String* s);

```

1 [VB] Public Shared Function ToChar(ByVal s As String) As Char

2 [JScript] public static function ToChar(s : String) : Char;

3  
4 *Description*

5 Converts the **System.String** to a **System.Char** equivalent.

6 *Return Value:* A **Char** representing the first character in the string. The string to  
7 convert.

8 *ToDateTime*

9  
10 [C#] public static DateTime ToDateTime(string s);

11 [C++] public: static DateTime ToDateTime(String\* s);

12 [VB] Public Shared Function ToDateTime(ByVal s As String) As DateTime

13 [JScript] public static function ToDateTime(s : String) : DateTime; Converts the  
14 **System.String** to a **System.DateTime** equivalent.

15  
16 *Description*

17 Converts the **System.String** to a **System.DateTime** equivalent.

18 *Return Value:* A **DateTime** equivalent of the string. The string to convert.

19 *ToDateTime*

20  
21 [C#] public static DateTime ToDateTime(string s, string format);

22 [C++] public: static DateTime ToDateTime(String\* s, String\* format);

23 [VB] Public Shared Function ToDateTime(ByVal s As String, ByVal format As  
24 String) As DateTime

25 [JScript] public static function ToDateTime(s : String, format : String) : DateTime;

## Description

Converts the **System.String** to a **System.DateTime** equivalent.

*Return Value:* A **DateTime** equivalent of the string. The string to convert. The format structure to apply to the converted **DateTime**. Valid formats include "yyyy-MM-ddTHH:mm:sszzzzzz" and its subsets. The string is validated against this format.

## DateTime

[C#] public static DateTime ToDateTime(string s, string[] formats);

[C++] public: static DateTime ToDateTime(String\* s, String\* formats \_\_gc[]);

[VB] Public Shared Function ToDateTime(ByVal s As String, ByVal formats() As String) As DateTime

[JScript] public static function ToDateTime(s : String, formats : String[]) :  
DateTime;

## Description

Converts the **System.String** to a **System.DateTime** equivalent.

*Return Value:* A **DateTime** equivalent of the string.

This method allows multiple formats for the string to be validated against. The string to convert. An array containing the format structures to apply to the converted **DateTime** . Valid formats include "yyyy-MM-ddTHH:mm:sszzzzzz" and its subsets.

## ToDecimal

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25

```
[C#] public static decimal ToDecimal(string s);
[C++] public: static Decimal ToDecimal(String* s);
[VB] Public Shared Function ToDecimal(ByVal s As String) As Decimal
[JScript] public static function ToDecimal(s : String) : Decimal;
```

*Description*

Converts the **System.String** to a **System.Decimal** equivalent.  
*Return Value:* A **Decimal** equivalent of the string. The string to convert.

**ToDouble**

```
[C#] public static double ToDouble(string s);
[C++] public: static double ToDouble(String* s);
[VB] Public Shared Function ToDouble(ByVal s As String) As Double
[JScript] public static function ToDouble(s : String) : double;
```

*Description*

Converts the **System.String** to a **System.Double** equivalent.  
*Return Value:* A **Double** equivalent of the string.  
 If *s* is INF or -INF, this method returns Double.PositiveInfinity or Double.NegativeInfinity respectively. The string to convert.

**ToGuid**

```
[C#] public static Guid ToGuid(string s);
[C++] public: static Guid ToGuid(String* s);
```

1 [VB] Public Shared Function ToGuid(ByVal s As String) As Guid

2 [JScript] public static function ToGuid(s : String) : Guid;

3  
4 *Description*

5 Converts the **System.String** to a **System.Guid** equivalent.

6 *Return Value:* A **Guid** equivalent of the string. The string to convert.

7 **ToInt16**

8  
9 [C#] public static short ToInt16(string s);

10 [C++] public: static short ToInt16(String\* s);

11 [VB] Public Shared Function ToInt16(ByVal s As String) As Short

12 [JScript] public static function ToInt16(s : String) : Int16;

13  
14 *Description*

15 Converts the **System.String** to a **System.Int16** equivalent.

16 *Return Value:* A **Int16** equivalent of the string. The string to convert.

17 **ToInt32**

18  
19 [C#] public static int ToInt32(string s);

20 [C++] public: static int ToInt32(String\* s);

21 [VB] Public Shared Function ToInt32(ByVal s As String) As Integer

22 [JScript] public static function ToInt32(s : String) : int;

23  
24 *Description*

Converts the **System.String** to a **System.Int32** equivalent.

*Return Value:* A **Int32** equivalent of the string. The string to convert.

ToInt64

[C#] public static long ToInt64(string s);

[C++] public: static \_\_int64 ToInt64(String\* s);

[VB] Public Shared Function ToInt64(ByVal s As String) As Long

[JScript] public static function ToInt64(s : String) : long;

#### *Description*

Converts the **System.String** to a **System.Int64** equivalent.

*Return Value:* A **Int64** equivalent of the string. The string to convert.

ToSByte

[C#] public static sbyte ToSByte(string s);

[C++] public: static char ToSByte(String\* s);

[VB] Public Shared Function ToSByte(ByVal s As String) As SByte

[JScript] public static function ToSByte(s : String) : SByte;

#### *Description*

Converts the **System.String** to a **System.SByte** equivalent.

*Return Value:* A **SByte** equivalent of the string. The string to convert.

ToSingle

[C#] public static float ToSingle(string s);

[C++] public: static float ToSingle(String\* s);

[VB] Public Shared Function ToSingle(ByVal s As String) As Single

[JScript] public static function ToSingle(s : String) : float;

### *Description*

Converts the **System.String** to a **System.Single** equivalent.

*Return Value:* A **Single** equivalent of the string.

If *s* is INF or -INF, this method returns Single.PositiveInfinity or Single.NegativeInfinity respectively. The string to convert.

### *ToString*

[C#] public static string ToString(bool value);

[C++] public: static String\* ToString(bool value);

[VB] Public Shared Function ToString(ByVal value As Boolean) As String

[JScript] public static function ToString(value : Boolean) : String; Converts strongly typed data to an equivalent **System.String** representation.

### *Description*

Converts the **System.Boolean** to a **System.String**.

*Return Value:* A string representation of the **Boolean** (i.e "true" or "false"). The value to convert.

### *ToString*

[C#] public static string ToString(byte value);

[C++] public: static String\* ToString(unsigned char value);

[VB] Public Shared Function ToString(ByVal value As Byte) As String

[JScript] public static function ToString(value : Byte) : String;

#### *Description*

Converts the **System.Byte** to a **System.String** .

*Return Value:* A string representation of the **Byte** . The value to convert.

ToString

[C#] public static string ToString(char value);

[C++] public: static String\* ToString(\_\_wchar\_t value);

[VB] Public Shared Function ToString(ByVal value As Char) As String

[JScript] public static function ToString(value : Char) : String;

#### *Description*

Converts the **System.Char** to a **System.String** .

*Return Value:* A string representation of the **Char** . The value to convert.

ToString

[C#] public static string ToString(DateTime value);

[C++] public: static String\* ToString(DateTime value);

[VB] Public Shared Function ToString(ByVal value As DateTime) As String

[JScript] public static function ToString(value : DateTime) : String;

#### *Description*



Converts the **System.DateTime** to a **System.String** .

*Return Value:* A string representation of the **DateTime** in the format yyyy-MM-ddTHH:mm:ss where 'T' is a constant literal. The value to convert.

ToString

[C#] public static string ToString(decimal value);

[C++] public: static String\* ToString(Decimal value);

[VB] Public Shared Function ToString(ByVal value As Decimal) As String

[JScript] public static function ToString(value : Decimal) : String;

#### *Description*

Converts the **System.Decimal** to a **System.String** .

*Return Value:* A string representation of the **Decimal** . The value to convert.

ToString

[C#] public static string ToString(double value);

[C++] public: static String\* ToString(double value);

[VB] Public Shared Function ToString(ByVal value As Double) As String

[JScript] public static function ToString(value : double) : String;

#### *Description*

Converts the **System.Double** to a **System.String** .

*Return Value:* A string representation of the **Double** .

If *value* is Double.PositiveInfinity or Double.NegativeInfinity, this method returns the string INF or -INF respectively. The value to convert.

## ToString

[C#] public static string ToString(Guid value);

[C++] public: static String\* ToString(Guid value);

[VB] Public Shared Function ToString(ByVal value As Guid) As String

[JScript] public static function ToString(value : Guid) : String;

### *Description*

Converts the **System.Guid** to a **System.String** .

*Return Value:* A string representation of the **Guid** . The value to convert.

## ToString

[C#] public static string ToString(short value);

[C++] public: static String\* ToString(short value);

[VB] Public Shared Function ToString(ByVal value As Short) As String

[JScript] public static function ToString(value : Int16) : String;

### *Description*

Converts the **System.Int16** to a **System.String** .

*Return Value:* A string representation of the **Int16** . The value to convert.

## ToString

[C#] public static string ToString(int value);

[C++] public: static String\* ToString(int value);

[VB] Public Shared Function ToString(ByVal value As Integer) As String

1 [JScript] public static function ToString(value : int) : String;

3 *Description*

4 Converts the **System.Int32** to a **System.String** .

5 *Return Value:* A string representation of the **Int32** . The value to convert.

6 ToString

8 [C#] public static string ToString(long value);

9 [C++] public: static String\* ToString(\_\_int64 value);

10 [VB] Public Shared Function ToString(ByVal value As Long) As String

11 [JScript] public static function ToString(value : long) : String;

13 *Description*

14 Converts the **System.Int64** to a **System.String** .

15 *Return Value:* A string representation of the **Int64** . The value to convert.

16 ToString

18 [C#] public static string ToString(sbyte value);

19 [C++] public: static String\* ToString(char value);

20 [VB] Public Shared Function ToString(ByVal value As SByte) As String

21 [JScript] public static function ToString(value : SByte) : String;

23 *Description*

24 Converts the **System.SByte** to a **System.String** .

25 *Return Value:* A string representation of the **SByte** . The value to convert.

## ToString

[C#] public static string ToString(float value);

[C++] public: static String\* ToString(float value);

[VB] Public Shared Function ToString(ByVal value As Single) As String

[JScript] public static function ToString(value : float) : String;

### *Description*

Converts the **System.Single** to a **System.String** .

*Return Value:* A string representation of the **Single** .

If *value* is Single.PositiveInfinity or Single.NegativeInfinity, this method returns the string INF or -INF respectively. The value to convert.

## ToString

[C#] public static string ToString(TimeSpan value);

[C++] public: static String\* ToString(TimeSpan value);

[VB] Public Shared Function ToString(ByVal value As TimeSpan) As String

[JScript] public static function ToString(value : TimeSpan) : String;

### *Description*

Converts the **System.TimeSpan** to a **System.String** .

*Return Value:* A string representation of the **TimeSpan** . The value to convert.

## ToString

[C#] public static string ToString(ushort value);

[C++] public: static String\* ToString(unsigned short value);

[VB] Public Shared Function ToString(ByVal value As UInt16) As String

[JScript] public static function ToString(value : UInt16) : String;

### *Description*

Converts the **System.UInt16** to a **System.String** .

*Return Value:* A string representation of the **UInt16** . The value to convert.

ToString

[C#] public static string ToString(uint value);

[C++] public: static String\* ToString(unsigned int value);

[VB] Public Shared Function ToString(ByVal value As UInt32) As String

[JScript] public static function ToString(value : UInt32) : String;

### *Description*

Converts the **System.UInt32** to a **System.String** .

*Return Value:* A string representation of the **UInt32** . The value to convert.

ToString

[C#] public static string ToString(ulong value);

[C++] public: static String\* ToString(unsigned \_\_int64 value);

[VB] Public Shared Function ToString(ByVal value As UInt64) As String

[JScript] public static function ToString(value : UInt64) : String;

### *Description*

Converts the **System.UInt64** to a **System.String** .

*Return Value:* A string representation of the **UInt64** . The value to convert.

**ToString**

[C#] public static string ToString(DateTime value, string format);

[C++] public: static String\* ToString(DateTime value, String\* format);

[VB] Public Shared Function ToString(ByVal value As DateTime, ByVal format As String) As String

[JScript] public static function ToString(value : DateTime, format : String) : String;

#### *Description*

Converts the **System.DateTime** to a **System.String** .

*Return Value:* A string representation of the **DateTime** in the specified format.

The value to convert. The format structure that defines how to display the converted string. Valid formats include "yyyy-MM-ddTHH:mm:sszzzzzz" and its subsets.

**ToTimeSpan**

[C#] public static TimeSpan ToTimeSpan(string s);

[C++] public: static TimeSpan ToTimeSpan(String\* s);

[VB] Public Shared Function ToTimeSpan(ByVal s As String) As TimeSpan

[JScript] public static function ToTimeSpan(s : String) : TimeSpan;

#### *Description*

Converts the **System.String** to a **System.TimeSpan** equivalent.

*Return Value:* A **TimeSpan** equivalent of the string. The string to convert.

ToUInt16

[C#] public static ushort ToUInt16(string s);

[C++] public: static unsigned short ToUInt16(String\* s);

[VB] Public Shared Function ToUInt16(ByVal s As String) As UInt16

[JScript] public static function ToUInt16(s : String) : UInt16;

#### Description

Converts the **System.String** to a **System.UInt16** equivalent.

*Return Value:* A **UInt16** equivalent of the string. The string to convert.

ToUInt32

[C#] public static uint ToUInt32(string s);

[C++] public: static unsigned int ToUInt32(String\* s);

[VB] Public Shared Function ToUInt32(ByVal s As String) As UInt32

[JScript] public static function ToUInt32(s : String) : UInt32;

#### Description

Converts the **System.String** to a **System.UInt32** equivalent.

*Return Value:* A **UInt32** equivalent of the string. The string to convert.

ToUInt64

[C#] public static ulong ToUInt64(string s);





1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25

*Description*

Attributes

BaseURI

ChildNodes

Encoding

ToUInt64

*Description*

Gets or sets the encoding level of the XML document.

Unlike most XML attributes, encoding attribute values are not case sensitive. This is because encoding character names follow ISO and Internet Assigned Numbers Authority (IANA) standards.

FirstChild

HasChildNodes

InnerText

ToUInt64

*Description*

Gets or sets the concatenated values of the **XmlDeclaration** .

InnerXml

IsReadOnly

Item

1 Item  
2 LastChild  
3 LocalName  
4 ToUInt64  
5  
6

7 *Description*

8 Gets the local name of the node.

9 Name  
10 ToUInt64  
11

12 [C#] public override string Name {get;}

13 [C++] public: \_\_property virtual String\* get\_Name();

14 [VB] Overrides Public ReadOnly Property Name As String

15 [JScript] public function get Name() : String;

16  
17 *Description*

18 Gets the qualified name of the node.

19 NamespaceURI  
20 NextSibling  
21 NodeType  
22 ToUInt64  
23  
24

25 *Description*

Gets the type of the current node.

OuterXml

OwnerDocument

ParentNode

Prefix

PreviousSibling

Standalone

ToUInt64

### *Description*

Gets or sets the value of the standalone attribute.

Value

ToUInt64

[C#] public override string Value {get; set;}

[C++] public: \_\_property virtual String\* get\_Value();public: \_\_property virtual  
void set\_Value(String\*);

[VB] Overrides Public Property Value As String

[JScript] public function get Value() : String;public function set Value(String);

### *Description*

Gets or sets the value of the **XmlDeclaration** .

Version

ToUInt64

```

1
2 [C#] public string Version {get;}
3 [C++] public: __property String* get_Version();
4 [VB] Public ReadOnly Property Version As String
5 [JScript] public function get Version() : String;
6

```

### *Description*

Gets the XML version of the document.

### **CloneNode**

```

11 [C#] public override XmlNode CloneNode(bool deep);
12 [C++] public: XmlNode* CloneNode(bool deep);
13 [VB] Overrides Public Function CloneNode(ByVal deep As Boolean) As
14 XmlNode
15 [JScript] public override function CloneNode(deep : Boolean) : XmlNode;
16

```

### *Description*

Creates a duplicate of this node.

*Return Value:* The cloned node.

**CloneNode** serves as a copy constructor for nodes. For **XmlDeclaration** nodes, the cloned node always includes the data value. To see how this method behaves with other node types, see **System.Xml.XmlNode.CloneNode(System.Boolean)** method in the **XmlNode** class. **true** to recursively clone the subtree under the specified node; **false** to clone only the node itself.

## WriteContentTo

[C#] public override void WriteContentTo(XmlWriter w);  
[C++] public: void WriteContentTo(XmlWriter\* w);  
[VB] Overrides Public Sub WriteContentTo(ByVal w As XmlWriter)  
[JScript] public override function WriteContentTo(w : XmlWriter);

### *Description*

Saves the children of the node to the specified **System.Xml.XmlWriter** .  
The **XmlWriter** to which you want to save.

## WriteTo

[C#] public override void WriteTo(XmlWriter w);  
[C++] public: void WriteTo(XmlWriter\* w);  
[VB] Overrides Public Sub WriteTo(ByVal w As XmlWriter)  
[JScript] public override function WriteTo(w : XmlWriter);

### *Description*

Saves the node to the specified **System.Xml.XmlWriter** . The **XmlWriter**  
to which you want to save.

XmlDocument class (System.Xml)

## WriteTo

### *Description*

Represents an XML document.

This class implements the W3C Document Object Model (DOM) Level 1 Core and the Core DOM Level 2. The DOM is an in-memory (cache) tree representation of an XML document and enables the navigation and editing of this document. Because **XmlDocument** implements the **System.Xml.XPath.IXPathNavigable** interface it can also be used as the source document for the **System.Xml.Xsl.XslTransform** class.

**XmlDocument**

*Example Syntax:*

**WriteTo**

[C#] public XmlDocument();

[C++] public: XmlDocument();

[VB] Public Sub New()

[JScript] public function XmlDocument(); Initializes a new instance of the **XmlDocument** class.

*Description*

Initializes a new instance of the **XmlDocument** class.

**XmlDocument**

*Example Syntax:*

**WriteTo**

[C#] protected internal XmlDocument(XmlImplementation imp);

[C++] internal: XmlDocument(XmlImplementation\* imp);

1 [VB] Protected Friend Sub New(ByVal imp As XmlImplementation)

2 [JScript] package function XmlDocument(imp : XmlImplementation);

3  
4 *Description*

5       Initializes a new instance of the **XmlDocument** class with the specified  
6 **System.Xml.XmlImplementation** . The **XmlImplementation** to use.

7       XmlDocument

8       *Example Syntax:*

9       WriteTo

10  
11 [C#] public XmlDocument(XmlNameTable nt);

12 [C++] public: XmlDocument(XmlNameTable\* nt);

13 [VB] Public Sub New(ByVal nt As XmlNameTable)

14 [JScript] public function XmlDocument(nt : XmlNameTable);

15  
16 *Description*

17       Initializes a new instance of the **XmlDocument** class with the specified  
18 **System.Xml.XmlNameTable** . The **XmlNameTable** to use.

19       Attributes

20       BaseURI

21       WriteTo

22  
23  
24 *Description*

25       Gets the base URI of the current node.

1 A networked XML document is comprised of chunks of data aggregated  
2 using various W3C standard inclusion mechanisms and therefore contains nodes  
3 that come from different places. The **BaseURI** tells you where these nodes came  
4 from.

5 ChildNodes

6 DocumentElement

7 WriteTo

8  
9  
10 *Description*

11 Gets the root **System.Xml.XmlElement** for the document.

12 DocumentType

13 WriteTo

14  
15 [C#] public virtual XmlDocumentType DocumentType {get;}

16 [C++] public: \_\_property virtual XmlDocumentType\* get\_DocumentType();

17 [VB] Overridable Public ReadOnly Property DocumentType As

18 XmlDocumentType

19 [JScript] public function get DocumentType() : XmlDocumentType;

20  
21 *Description*

22 Gets the node containing the DOCTYPE declaration.

23 An **XmlDocument** may have one and only one child with

24 **System.Xml.XmlNodeType** equal to **DocumentType**.

25 FirstChild



HasChildNodes

Implementation

WriteTo

*Description*

Gets the **System.Xml.XmlImplementation** object for the current document.

**XmlDocument** objects created from the same **XmlImplementation** share the same **System.Xml.XmlNameTable** . This allows users to compare attribute and element names as objects rather than strings.

InnerText

InnerXml

WriteTo

*Description*

Gets or sets the markup representing just the children of this node.

This is also settable which replaces the children of the node with the parsed contents of the given string. The parsing is done in the current namespace context.

IsReadOnly

WriteTo

[C#] public override bool IsReadOnly {get;}

[C++] public: \_\_property virtual bool get\_IsReadOnly();

1 [VB] Overrides Public ReadOnly Property IsReadOnly As Boolean

2 [JScript] public function get IsReadOnly() : Boolean;

3  
4 *Description*

5 Gets a value indicating whether the current node is read-only.

6 A read-only node is one whose properties, attributes, or children cannot be  
7 changed. You can remove a read-only node from the tree and insert it somewhere  
8 else.

9 Item

10 Item

11 LastChild

12 LocalName

13 WriteTo

14  
15  
16 *Description*

17 Gets the local name of the node.

18 The local name returned depends on the  
19 **System.Xml.XmlDocument.NodeType** of the node. To see a table describing the  
20 local name returned for each of the node types, see the  
21 **System.Xml.XmlNode.LocalName** property in the **System.Xml.XmlNode** class.

22 Name

23 WriteTo

24  
25 [C#] public override string Name {get;}

```
1 [C++] public: __property virtual String* get_Name();
2 [VB] Overrides Public ReadOnly Property Name As String
3 [JScript] public function get Name() : String;
```

#### *Description*

Gets the qualified name of the node.

The name returned depends on the **System.Xml.XmlDocument.NodeType** of the node. To see a table describing the name returned for each of the node types, see the **System.Xml.XmlNode.Name** property in the **System.Xml.XmlNode** class.

NamespaceURI

NameTable

WriteTo

#### *Description*

Gets the **System.Xml.XmlNameTable** associated with this implementation.

Each **XmlDocument** object has a **System.Xml.NameTable** object. Element and attribute names are stored in the **NameTable** as atomized strings. This means that even if a name is referenced in the document multiple times it is stored only once in the **NameTable**. For example, if the document had multiple elements with the name "Customer", **NameTable** returns the same object whenever it receives a request for that name. As a result, users can write code

using object pointer comparisons on these strings rather than the more expensive string comparisons.

NextSibling

NodeType

WriteTo

### *Description*

Gets the type of the current node.

OuterXml

OwnerDocument

WriteTo

### *Description*

Gets the **System.Xml.XmlDocument** to which the current node belongs.

ParentNode

Prefix

PreserveWhitespace

WriteTo

### *Description*

Gets or sets a value indicating whether to preserve whitespace.

PreviousSibling

Value

XmlResolver

WriteTo

#### *Description*

Sets the **System.Xml.XmlResolver** to use for resolving external resources.

The **XmlResolver** can be used to load DTDs or expand entity references.

Using the **System.Xml.XmlResolver.Credentials** property, you can set credentials on the **XmlResolver** to access resources stored on a secure network resource.

WriteTo

[C#] public event XmlNodeChangedEventHandler NodeChanged;

[C++] public: \_\_event XmlNodeChangedEventHandler\* NodeChanged;

[VB] Public Event NodeChanged As XmlNodeChangedEventHandler

#### *Description*

Occurs when the **System.Xml.XmlNode.Value** of a node belonging to this document has been changed.

This event only applies to nodes that have a value.

WriteTo

[C#] public event XmlNodeChangedEventHandler NodeChanging;

[C++] public: \_\_event XmlNodeChangedEventHandler\* NodeChanging;

1 [VB] Public Event NodeChanging As XmlNodeChangedEventHandler

3 *Description*

4 Occurs when the **System.Xml.XmlNode.Value** of a node belonging to this  
5 document is about to be changed.

6 This event allows the user to do extra checking and, if necessary, throw an  
7 exception to stop the operation. If an exception is thrown the **XmlDocument**  
8 returns to its original state. This event only applies to nodes that have a value.

9 WriteTo

11 [C#] public event XmlNodeChangedEventHandler NodeInserted;

12 [C++] public: \_\_event XmlNodeChangedEventHandler\* NodeInserted;

13 [VB] Public Event NodeInserted As XmlNodeChangedEventHandler

15 *Description*

16 Occurs when a node belonging to this document has been inserted into  
17 another node.

18 All nodes created by this document, whether or not they have been inserted  
19 into the document, are included in this event.

20 WriteTo

22 [C#] public event XmlNodeChangedEventHandler NodeInserting;

23 [C++] public: \_\_event XmlNodeChangedEventHandler\* NodeInserting;

24 [VB] Public Event NodeInserting As XmlNodeChangedEventHandler

1  
2 *Description*

3 Occurs when a node belonging to this document is about to be inserted into  
4 another node.

5 This event allows the user to do extra checking and, if necessary, throw an  
6 exception to stop the operation. If an exception is thrown the **XmlDocument**  
7 returns to its original state.

8 WriteTo

9  
10 [C#] public event XmlNodeChangedEventHandler NodeRemoved;

11 [C++] public: \_\_event XmlNodeChangedEventHandler\* NodeRemoved;

12 [VB] Public Event NodeRemoved As XmlNodeChangedEventHandler

13  
14 *Description*

15 Occurs when a node belonging to this document has been removed from its  
16 parent.

17 All nodes created by this document, whether or not they have been inserted  
18 into the document, are included in this event.

19 WriteTo

20  
21 [C#] public event XmlNodeChangedEventHandler NodeRemoving;

22 [C++] public: \_\_event XmlNodeChangedEventHandler\* NodeRemoving;

23 [VB] Public Event NodeRemoving As XmlNodeChangedEventHandler

24  
25 *Description*

Occurs when a node belonging to this document is about to be removed from the document.

This event allows the user to do extra checking and, if necessary, throw an exception to stop the operation. If an exception is thrown the **XmlDocument** returns to its original state.

## CloneNode

[C#] public override XmlNode CloneNode(bool deep);

[C++] public: XmlNode\* CloneNode(bool deep);

[VB] Overrides Public Function CloneNode(ByVal deep As Boolean) As XmlNode

[JScript] public override function CloneNode(deep : Boolean) : XmlNode;

## Description

Creates a duplicate of this node.

*Return Value:* The cloned **XmlDocument** node.

This method serves as a copy constructor for nodes. The cloned node has no parent ( **System.Xml.XmlNode.ParentNode** returns **null** ). **true** to recursively clone the subtree under the specified node; **false** to clone only the node itself.

## CreateAttribute

[C#] public XmlAttribute CreateAttribute(string name);

[C++] public: XmlAttribute\* CreateAttribute(String\* name);

[VB] Public Function CreateAttribute(ByVal name As String) As XmlAttribute

[JScript] public function CreateAttribute(name : String) : XmlAttribute; Creates an



**System.Xml.XmlAttribute** with the specified name.

### *Description*

Creates an **System.Xml.XmlAttribute** with the specified

**System.Xml.XmlDocument.Name** .

*Return Value:* The new **XmlAttribute** .

The **XmlAttribute** can be added to an **System.Xml.XmlElement** using the

**System.Xml.XmlElement.SetAttributeNode(System.Xml.XmlAttribute)**

method. The qualified name of the attribute. If the name contains a colon, the

**System.Xml.XmlNode.Prefix** property reflects the part of the name preceding the

first colon and the **System.Xml.XmlDocument.LocalName** property reflects the

part of the name following the first colon. The

**System.Xml.XmlNode.NamespaceURI** remains empty unless the prefix is a

recognized built-in prefix such as xmlns. In this case **NamespaceURI** has a value

of <http://www.w3.org/2000/xmlns/> .

### **CreateAttribute**

```
[C#] public XmlAttribute CreateAttribute(string qualifiedName, string  
namespaceURI);
```

```
[C++] public: XmlAttribute* CreateAttribute(String* qualifiedName, String*  
namespaceURI);
```

```
[VB] Public Function CreateAttribute(ByVal qualifiedName As String, ByVal  
namespaceURI As String) As XmlAttribute
```

```
[JScript] public function CreateAttribute(qualifiedName : String, namespaceURI :  
String) : XmlAttribute;
```

## Description

Creates an **System.Xml.XmlAttribute** with the specified qualified name and **System.Xml.XmlNode.NamespaceURI** .

*Return Value:* The new **XmlAttribute** .

The **XmlAttribute** can be added to an **System.Xml.XmlElement** using the **System.Xml.XmlElement.SetAttributeNode(System.Xml.XmlAttribute)** method. The qualified name of the attribute. If the name contains a colon then the **System.Xml.XmlNode.Prefix** property will reflect the part of the name preceding the colon and the **System.Xml.XmlDocument.LocalName** property will reflect the part of the name after the colon. The namespaceURI of the attribute. If the qualified name includes a prefix of xmlns, then this parameter must be <http://www.w3.org/2000/xmlns/> .

## CreateAttribute

[C#] public virtual XmlAttribute CreateAttribute(string prefix, string localName, string namespaceURI);

[C++] public: virtual XmlAttribute\* CreateAttribute(String\* prefix, String\* localName, String\* namespaceURI);

[VB] Overridable Public Function CreateAttribute(ByVal prefix As String, ByVal localName As String, ByVal namespaceURI As String) As XmlAttribute

[JScript] public function CreateAttribute(prefix : String, localName : String, namespaceURI : String) : XmlAttribute;

## Description

Creates an **System.Xml.XmlAttribute** with the specified **System.Xml.XmlNode.Prefix** , **System.Xml.XmlDocument.LocalName** , and **System.Xml.XmlNode.NamespaceURI** .

*Return Value:* The new **XmlAttribute** .

The **XmlAttribute** can be added to an **System.Xml.XmlElement** using the **System.Xml.XmlElement.SetAttributeNode(System.Xml.XmlAttribute)** method. The prefix of the attribute (if any). **String.Empty** and **null** are equivalent. The local name of the attribute. The namespace URI of the attribute (if any). **String.Empty** and **null** are equivalent. If *prefix* is *xmlns*, then this parameter must be <http://www.w3.org/2000/xmlns/>; otherwise an exception is thrown.

#### CreateCDATASection

[C#] public virtual XmlCDATASection CreateCDATASection(string data);  
 [C++] public: virtual XmlCDATASection\* CreateCDATASection(String\* data);  
 [VB] Overridable Public Function CreateCDATASection(ByVal data As String) As XmlCDATASection  
 [JScript] public function CreateCDATASection(data : String) : XmlCDATASection;

#### Description

Creates an **System.Xml.XmlCDATASection** containing the specified data.

*Return Value:* The new **XmlCDATASection** .

Although this method creates the new object in the context of the document, it does not automatically add the new object to the document tree. To add the new object, you must explicitly call one of the node insert methods. The content of the new **XmlCDATASection** .

## CreateComment

```
[C#] public virtual XmlComment CreateComment(string data);
[C++] public: virtual XmlComment* CreateComment(String* data);
[VB] Overridable Public Function CreateComment(ByVal data As String) As
```

XmlComment

```
[JScript] public function CreateComment(data : String) : XmlComment;
```

### *Description*

Creates an **System.Xml.XmlComment** containing the specified data.

*Return Value:* The new **XmlComment** .

Although this method creates the new object in the context of the document, it does not automatically add the new object to the document tree. To add the new object, you must explicitly call one of the node insert methods. The content of the new **XmlComment** .

## CreateDefaultAttribute

```
[C#] protected internal virtual XmlAttribute CreateDefaultAttribute(string prefix,
string localName, string namespaceURI);
```

```
[C++] protected public: virtual XmlAttribute* CreateDefaultAttribute(String*
prefix, String* localName, String* namespaceURI);
```

```
[VB] Overridable Protected Friend Dim Function CreateDefaultAttribute(ByVal
prefix As String, ByVal localName As String, ByVal namespaceURI As String)
```

As XmlAttribute

```
[JScript] package function CreateDefaultAttribute(prefix : String, localName :
```

String, namespaceURI : String) : XmlAttribute;

### *Description*

Creates a default attribute with the specified prefix, local name and namespace URI.

*Return Value:* The new **System.Xml.XmlAttribute** . The prefix of the attribute (if any). The local name of the attribute. The namespace URI of the attribute (if any).

### **CreateDocumentFragment**

[C#] public virtual XmlDocumentFragment CreateDocumentFragment();

[C++] public: virtual XmlDocumentFragment\* CreateDocumentFragment();

[VB] Overridable Public Function CreateDocumentFragment() As

XmlDocumentFragment

[JScript] public function CreateDocumentFragment() : XmlDocumentFragment;

### *Description*

Creates an **System.Xml.XmlDocumentFragment** .

*Return Value:* The new **XmlDocumentFragment** .

DocumentFragment nodes cannot be inserted into a document. However, you can insert children of the DocumentFragment node into a document.

### **CreateDocumentType**

[C#] public virtual XmlDocumentType CreateDocumentType(string name, string

publicId, string systemId, string internalSubset);

[C++] public: virtual XmlDocumentType\* CreateDocumentType(String\* name,

```

1 String* publicId, String* systemId, String* internalSubset);
2 [VB] Overridable Public Function CreateDocumentType(ByVal name As String,
3 ByVal publicId As String, ByVal systemId As String, ByVal internalSubset As
4 String) As XmlDocumentType
5 [JScript] public function CreateDocumentType(name : String, publicId : String,
6 systemId : String, internalSubset : String) : XmlDocumentType;

```

#### *Description*

Returns a new **System.Xml.XmlDocumentType** object.

*Return Value:* The new **XmlDocumentType** .

The returned node will have parsed

**System.Xml.XmlDocumentType.Entities** and

**System.Xml.XmlDocumentType.Notations** collections. Name of the document type. The public identifier of the document type or **null** . The system identifier of the document type or **null** . The DTD internal subset of the document type or **null**

.

#### *CreateElement*

```

19 [C#] public XmlElement CreateElement(string name);
20 [C++] public: XmlElement* CreateElement(String* name);
21 [VB] Public Function CreateElement(ByVal name As String) As XmlElement
22 [JScript] public function CreateElement(name : String) : XmlElement; Creates an
23 System.Xml.XmlElement .

```

#### *Description*

Creates an element with the specified name.

*Return Value:* The new **XmlElement** .

Note that the instance returned implements the **XmlElement** interface, so default attributes would be created directly on the returned object. The qualified name of the element. If the name contains a colon then the **System.Xml.XmlNode.Prefix** property reflects the part of the name preceding the colon and the **System.Xml.XmlDocument.LocalName** property reflects the part of the name after the colon. The qualified name cannot include a prefix of 'xmlns'.

### CreateElement

```

[C#] public XmlElement CreateElement(string qualifiedName, string
namespaceURI);
[C++] public: XmlElement* CreateElement(String* qualifiedName, String*
namespaceURI);
[VB] Public Function CreateElement(ByVal qualifiedName As String, ByVal
namespaceURI As String) As XmlElement
[JScript] public function CreateElement(qualifiedName : String, namespaceURI :
String) : XmlElement;
    
```

### Description

Creates an **System.Xml.XmlElement** with the qualified name and **System.Xml.XmlNode.NamespaceURI** .

*Return Value:* The new **XmlElement** .

The following C# code XmlElement elem;  
elem=doc.CreateElement("item:bar", "urn:abc"); results in an XmlElement that is

equivalent to the following XML text. The qualified name of the element. If the name contains a colon then the **System.Xml.XmlNode.Prefix** property will reflect the part of the name preceding the colon and the **System.Xml.XmlDocument.LocalName** property will reflect the part of the name after the colon. The qualified name cannot include a prefix of 'xmlns'. The namespace URI of the element.

### CreateElement

[C#] public virtual XmlElement CreateElement(string prefix, string localName, string namespaceURI);

[C++] public: virtual XmlElement\* CreateElement(String\* prefix, String\* localName, String\* namespaceURI);

[VB] Overridable Public Function CreateElement(ByVal prefix As String, ByVal localName As String, ByVal namespaceURI As String) As XmlElement

[JScript] public function CreateElement(prefix : String, localName : String, namespaceURI : String) : XmlElement;

### Description

Creates an element with the specified **System.Xml.XmlNode.Prefix** , **System.Xml.XmlDocument.LocalName** , and **System.Xml.XmlNode.NamespaceURI** .

*Return Value:* The new **System.Xml.XmlElement** .

The following C# code XmlElement elem;  
elem=doc.CreateElement("item", "bar", "urn:abc"); creates an element equivalent to the following XML text: Although this method creates the new object in the



context of the document, it does not automatically add the new object to the document tree. To add the new object, you must explicitly call one of the node insert methods. The prefix of the new element (if any). String.Empty and **null** are equivalent. The local name of the new element. The namespace URI of the new element (if any). String.Empty and **null** are equivalent.

## CreateEntityReference

```
[C#] public virtual XmlEntityReference CreateEntityReference(string name);
[C++] public: virtual XmlEntityReference* CreateEntityReference(String* name);
[VB] Overridable Public Function CreateEntityReference(ByVal name As String)
As XmlEntityReference
[JScript] public function CreateEntityReference(name : String) :
XmlEntityReference;
```

## Description

Creates an **System.Xml.XmlEntityReference** with the specified name.

**Return Value:** The new **XmlEntityReference**.

If the referenced entity is known, the child list of the **XmlEntityReference** node is made the same as that of the corresponding **System.Xml.XmlEntity** node. The name of the entity reference.

## CreateNavigator

```
[C#] protected internal virtual XPathNavigator CreateNavigator(XmlNode node);
[C++] protected public: virtual XPathNavigator* CreateNavigator(XmlNode*
node);
```

1 [VB] Overridable Protected Friend Dim Function CreateNavigator(ByVal node As  
2 XmlNode) As XPathNavigator

3 [JScript] package function CreateNavigator(node : XmlNode) : XPathNavigator;

4 Creates a new **System.Xml.XPath.XPathNavigator** object for navigating this  
5 document.

### 6 *Description*

7  
8 Creates an **System.Xml.XPath.XPathNavigator** object for navigating this  
9 document.

10 *Return Value:* An **System.Xml.XPath.XPathNavigator** object. The  
11 **System.Xml.XmlNode** to navigate.

### 12 *CreateNode*

13  
14 [C#] public virtual XmlNode CreateNode(string nodeTypeString, string name,  
15 string namespaceURI);

16 [C++] public: virtual XmlNode\* CreateNode(String\* nodeTypeString, String\*  
17 name, String\* namespaceURI);

18 [VB] Overridable Public Function CreateNode(ByVal nodeTypeString As String,  
19 ByVal name As String, ByVal namespaceURI As String) As XmlNode

20 [JScript] public function CreateNode(nodeTypeString : String, name : String,  
21 namespaceURI : String) : XmlNode;

### 22 *Description*

23  
24 Creates an **System.Xml.XmlNode** with the specified node type,  
25 **System.Xml.XmlDocument.Name** , and **System.Xml.XmlNode.NamespaceURI**

*Return Value:* The new **XmlNode** .

The *nodeTypeString* is case sensitive and must be one of the following:

string XmlNodeType element Element attribute Attribute text Text cdatasection  
CDATA entityreference EntityReference processinginstruction  
ProcessingInstruction comment Comment document Document documenttype  
DocumentType documentfragment DocumentFragment Although this method  
creates the new object in the context of the document, it does not automatically  
add the new object to the document tree. To add the new object, you must  
explicitly call one of the node insert methods. String version of the  
**System.Xml.XmlNodeType** of the new node. This parameter must be one of the  
values listed below. The qualified name of the new node. If the name contains a  
colon, it is parsed into **System.Xml.XmlNode.Prefix** and  
**System.Xml.XmlDocument.LocalName** components. The namespace URI of the  
new node.

CreateNode

[C#] public virtual XmlNode CreateNode(XmlNodeType type, string name, string  
namespaceURI);

[C++] public: virtual XmlNode\* CreateNode(XmlNodeType type, String\* name,  
String\* namespaceURI);

[VB] Overridable Public Function CreateNode(ByVal type As XmlNodeType,  
ByVal name As String, ByVal namespaceURI As String) As XmlNode

[JScript] public function CreateNode(type : XmlNodeType, name : String,  
namespaceURI : String) : XmlNode;

## *Description*

Creates an XmlNode with the specified **System.Xml.XmlNodeType** ,  
**System.Xml.XmlDocument.Name** , and **System.Xml.XmlNode.NamespaceURI**

*Return Value:* The new **XmlNode** .

Although this method creates the new object in the context of the document, it does not automatically add the new object to the document tree. To add the new object, you must explicitly call one of the node insert methods. The **XmlNodeType** of the new node. The qualified name of the new node. If the name contains a colon then it is parsed into **System.Xml.XmlNode.Prefix** and **System.Xml.XmlDocument.LocalName** components. The namespace URI of the new node.

## **CreateNode**

[C#] public virtual XmlNode CreateNode(XmlNodeType type, string prefix, string name, string namespaceURI);

[C++] public: virtual XmlNode\* CreateNode(XmlNodeType type, String\* prefix, String\* name, String\* namespaceURI);

[VB] Overridable Public Function CreateNode(ByVal type As XmlNodeType, ByVal prefix As String, ByVal name As String, ByVal namespaceURI As String) As XmlNode

[JScript] public function CreateNode(type : XmlNodeType, prefix : String, name : String, namespaceURI : String) : XmlNode; Creates an **System.Xml.XmlNode** .

1  
2 *Description*

3       Creates a **System.Xml.XmlNode** with the specified  
4 **System.Xml.XmlNodeType** , **System.Xml.XmlNode.Prefix** ,  
5 **System.Xml.XmlDocument.Name** , and **System.Xml.XmlNode.NamespaceURI**  
6 .

7 *Return Value:* The new **XmlNode** .

8       Although this method creates the new object in the context of the  
9 document, it does not automatically add the new object to the document tree. To  
10 add the new object, you must explicitly call one of the node insert methods. The  
11 **XmlNodeType** of the new node. The prefix of the new node. The local name of  
12 the new node. The namespace URI of the new node.

13       CreateProcessingInstruction

14  
15 [C#] public virtual XmlProcessingInstruction CreateProcessingInstruction(string  
16 target, string data);

17 [C++] public: virtual XmlProcessingInstruction\*  
18 CreateProcessingInstruction(String\* target, String\* data);

19 [VB] Overridable Public Function CreateProcessingInstruction(ByVal target As  
20 String, ByVal data As String) As XmlProcessingInstruction

21 [JScript] public function CreateProcessingInstruction(target : String, data : String)  
22 : XmlProcessingInstruction;

23  
24 *Description*  
25

Creates an **System.Xml.XmlProcessingInstruction** with the specified name and data.

*Return Value:* The new **XmlProcessingInstruction** .

Although this method creates the new object in the context of the document, it does not automatically add the new object to the document tree. To add the new object, you must explicitly call one of the node insert methods. The name of the processing instruction. The data for the processing instruction.

#### CreateSignificantWhitespace

[C#] public virtual XmlSignificantWhitespace CreateSignificantWhitespace(string text);

[C++] public: virtual XmlSignificantWhitespace\* CreateSignificantWhitespace(String\* text);

[VB] Overridable Public Function CreateSignificantWhitespace(ByVal text As String) As XmlSignificantWhitespace

[JScript] public function CreateSignificantWhitespace(text : String) : XmlSignificantWhitespace;

#### Description

Creates an **System.Xml.XmlSignificantWhitespace** node.

*Return Value:* A new **XmlSignificantWhitespace** node.

This is used when you want to manually format your document. The string must contain only the following characters and

#### CreateTextNode

```

1 [C#] public virtual XmlText CreateTextNode(string text);
2
3 [C++] public: virtual XmlText* CreateTextNode(String* text);
4
5 [VB] Overridable Public Function CreateTextNode(ByVal text As String) As
6 XmlText
7
8 [JScript] public function CreateTextNode(text : String) : XmlText;
9

```

### *Description*

Creates an **System.Xml.XmlText** with the specified text.

*Return Value:* The new **XmlText** node.

Although this method creates the new object in the context of the document, it does not automatically add the new object to the document tree. To add the new object, you must explicitly call one of the node insert methods. The text for the Text node.

### *CreateWhitespace*

```

17 [C#] public virtual XmlWhitespace CreateWhitespace(string text);
18
19 [C++] public: virtual XmlWhitespace* CreateWhitespace(String* text);
20
21 [VB] Overridable Public Function CreateWhitespace(ByVal text As String) As
22 XmlWhitespace
23
24 [JScript] public function CreateWhitespace(text : String) : XmlWhitespace;
25

```

### *Description*

Creates an **System.Xml.XmlWhitespace** node.

*Return Value:* A new **XmlWhitespace** node.

This is used when you want to manually format your document. The string must contain only the following characters and

CreateXmlDeclaration

[C#] public virtual XmlDeclaration CreateXmlDeclaration(string version, string encoding, string standalone);

[C++] public: virtual XmlDeclaration\* CreateXmlDeclaration(String\* version, String\* encoding, String\* standalone);

[VB] Overridable Public Function CreateXmlDeclaration(ByVal version As String, ByVal encoding As String, ByVal standalone As String) As XmlDeclaration

[JScript] public function CreateXmlDeclaration(version : String, encoding : String, standalone : String) : XmlDeclaration;

### *Description*

Creates an **System.Xml.XmlDeclaration** node with the specified values.

*Return Value:* The new **XmlDeclaration** node.

The attributes are exposed as special properties on the **XmlDeclaration** node, and not as **System.Xml.XmlAttribute** nodes. The version must be "1.0".

The value of the encoding attribute. This is the encoding that is used when you save the **System.Xml.XmlDocument**; therefore, it must be set to a string supported by the **System.Text.Encoding** class, otherwise

**System.Xml.XmlDocument.Save(System.String)** fails. If this is **null** or

**String.Empty**, the **Save** method does not write an encoding attribute on the XML

declaration and therefore the default encoding, UTF-8, is used. The value must be



either "yes" or "no". If this is **null** or `String.Empty`, the **Save** method does not write a standalone attribute on the XML declaration.

### GetElementById

[C#] public virtual XmlElement GetElementById(string elementId);  
[C++] public: virtual XmlElement\* GetElementById(String\* elementId);  
[VB] Overridable Public Function GetElementById(ByVal elementId As String)  
As XmlElement  
[JScript] public function GetElementById(elementId : String) : XmlElement;

### *Description*

Gets the **System.Xml.XmlElement** with the specified ID.

*Return Value:* The **XmlElement** with the matching ID or **null** if no matching element is found.

If the document has multiple elements with the matching ID, this method returns the first matching element in the document. The attribute ID to match.

### GetElementsByTagName

[C#] public virtual XmlNodeList GetElementsByTagName(string name);  
[C++] public: virtual XmlNodeList\* GetElementsByTagName(String\* name);  
[VB] Overridable Public Function GetElementsByTagName(ByVal name As String) As XmlNodeList  
[JScript] public function GetElementsByTagName(name : String) : XmlNodeList;

Returns an **System.Xml.XmlNodeList** containing a list of all descendant elements that match the specified name.

## Description

Returns an **System.Xml.XmlNodeList** containing a list of all descendant elements that match the specified **System.Xml.XmlDocument.Name** .

*Return Value:* An **System.Xml.XmlNodeList** containing a list of all matching nodes.

The nodes are placed in the order in which they would be encountered in the document. The qualified name to match. It is matched against the **Name** property of the matching node. The special value "\*" matches all tags.

## GetElementsByTagName

[C#] public virtual XmlNodeList GetElementsByTagName(string localName, string namespaceURI);

[C++] public: virtual XmlNodeList\* GetElementsByTagName(String\* localName, String\* namespaceURI);

[VB] Overridable Public Function GetElementsByTagName(ByVal localName As String, ByVal namespaceURI As String) As XmlNodeList

[JScript] public function GetElementsByTagName(localName : String, namespaceURI : String) : XmlNodeList;

## Description

Returns an **System.Xml.XmlNodeList** containing a list of all descendant elements that match the specified **System.Xml.XmlDocument.LocalName** and **System.Xml.XmlNode.NamespaceURI** .

*Return Value:* An **System.Xml.XmlNodeList** containing a list of all matching nodes.

The nodes are placed in the order in which they would be encountered in the document tree. The LocalName to match. The special value "\*" matches all tags. NamespaceURI to match.

## ImportNode

[C#] public virtual XmlNode ImportNode(XmlNode node, bool deep);

[C++] public: virtual XmlNode\* ImportNode(XmlNode\* node, bool deep);

[VB] Overridable Public Function ImportNode(ByVal node As XmlNode, ByVal deep As Boolean) As XmlNode

[JScript] public function ImportNode(node : XmlNode, deep : Boolean) :

XmlNode;

## Description

Imports a node from another document to the current document.

*Return Value:* The imported **System.Xml.XmlNode** .

The returned node has no parent. The source node is not altered or removed from the original document; **ImportNode** creates a copy of the source node. The node being imported. **true** to perform a deep clone; otherwise, **false** .

## Load

[C#] public virtual void Load(Stream inStream);

[C++] public: virtual void Load(Stream\* inStream);

[VB] Overridable Public Sub Load(ByVal inStream As Stream)

1 [JScript] public function Load(inStream : Stream);

3 *Description*

4 Loads the XML document from the specified stream.

5 **System.Xml.XmlWhitespace** nodes are not created unless  
6 **System.Xml.XmlDocument.PreserveWhitespace** is set to **true** . Also, if the  
7 reader has been configured to not return whitespace, then no whitespace nodes  
8 will be created. In other words **Load** does not change the whitespace handling of  
9 the given reader. The stream containing the XML document to load.

10 Load

12 [C#] public virtual void Load(string filename);

13 [C++] public: virtual void Load(String\* filename);

14 [VB] Overridable Public Sub Load(ByVal filename As String)

15 [JScript] public function Load(filename : String); Loads the specified XML data.

17 *Description*

18 Loads the XML document from the specified URL.

19 **System.Xml.XmlWhitespace** nodes are not created unless  
20 **System.Xml.XmlDocument.PreserveWhitespace** is set to **true** . URL for the file  
21 containing the XML document to load.

22 Load

24 [C#] public virtual void Load(Reader txtReader);

25 [C++] public: virtual void Load(Reader\* txtReader);

[VB] Overridable Public Sub Load(ByVal txtReader As TextReader)

[JScript] public function Load(txtReader : TextReader);

#### *Description*

Loads the XML document from the specified **System.IO.TextReader** .

**System.Xml.XmlWhitespace** nodes are not created unless

**System.Xml.XmlDocument.PreserveWhitespace** is set to **true** . The

**TextReader** used to feed the XML data into the document.

Load

[C#] public virtual void Load(XmlReader reader);

[C++] public: virtual void Load(XmlReader\* reader);

[VB] Overridable Public Sub Load(ByVal reader As XmlReader)

[JScript] public function Load(reader : XmlReader);

#### *Description*

Loads the XML document from the specified **System.Xml.XmlReader** .

Whitespace nodes are not created unless

**System.Xml.XmlDocument.PreserveWhitespace** is set to **true** . Also, if the reader has been configured to not return whitespace, then no whitespace nodes will be created. In other words **Load** does not change the whitespace handling of the given reader. The **XmlReader** used to feed the XML data into the document.

LoadXml

[C#] public virtual void LoadXml(string xml);

1 [C++] public: virtual void LoadXml(String\* xml);

2 [VB] Overridable Public Sub LoadXml(ByVal xml As String)

3 [JScript] public function LoadXml(xml : String);

4  
5 *Description*

6 Loads the XML document from the specified string.

7 By default the **LoadXml** method does not preserve whitespace nor  
8 significant whitespace. String containing the XML document to load.

9 **ReadNode**

10  
11 [C#] public virtual XmlNode ReadNode(XmlReader reader);

12 [C++] public: virtual XmlNode\* ReadNode(XmlReader\* reader);

13 [VB] Overridable Public Function ReadNode(ByVal reader As XmlReader) As  
14 XmlNode

15 [JScript] public function ReadNode(reader : XmlReader) : XmlNode;

16  
17 *Description*

18 Creates an **System.Xml.XmlNode** object based on the information in the  
19 **System.Xml.XmlReader** . The reader must be positioned on a node or attribute.

20 *Return Value:* The new **XmlNode** or **null** if no more nodes exist.

21 Reads one **XmlNode** from the given reader and positions the reader on the  
22 next node. This method creates the type of **XmlNode** matching the  
23 **System.Xml.XmlNode.NodeType** on which the reader is currently positioned. (If  
24 the reader is in the initial state, **ReadNode** advances the reader to the first node  
25 and then operates on that node.) If the reader is positioned on the start of an

1 element, **ReadNode** reads all the attributes and any child nodes, up to and  
2 including the end tag of the current node. The **XmlNode** returned contains the  
3 subtree representing everything read. The reader is positioned immediately after  
4 the end tag. The Xml source

#### 5 Save

6  
7 [C#] public virtual void Save(Stream outputStream);

8 [C++] public: virtual void Save(Stream\* outputStream);

9 [VB] Overridable Public Sub Save(ByVal outputStream As Stream)

10 [JScript] public function Save(outputStream : Stream);

#### 11 12 *Description*

13 Saves the XML document to the specified stream.

14 Whitespace is preserved only if

15 **System.Xml.XmlDocument.PreserveWhitespace** is set to **true** . The stream to  
16 which you want to save.

#### 17 Save

18  
19 [C#] public virtual void Save(string filename);

20 [C++] public: virtual void Save(String\* filename);

21 [VB] Overridable Public Sub Save(ByVal filename As String)

22 [JScript] public function Save(filename : String); Saves the XML document to the  
23 specified location.

#### 24 25 *Description*

Saves the XML document to the specified file.

Whitespace is preserved in the output file only if

**System.Xml.XmlDocument.PreserveWhitespace** is set to **true** . The location of the file where you want to save the document.

Save

[C#] public virtual void Save(TextWriter writer);

[C++] public: virtual void Save(TextWriter\* writer);

[VB] Overridable Public Sub Save(ByVal writer As TextWriter)

[JScript] public function Save(writer : TextWriter);

### *Description*

Saves the XML document to the specified **System.IO.TextWriter** . The **TextWriter** to which you want to save.

Save

[C#] public virtual void Save(XmlWriter writer);

[C++] public: virtual void Save(XmlWriter\* writer);

[VB] Overridable Public Sub Save(ByVal writer As XmlWriter)

[JScript] public function Save(writer : XmlWriter);

### *Description*

Saves the XML document to the specified **System.Xml.XmlWriter** .



1        Whitespace is preserved only if  
 2        **System.Xml.XmlDocument.PreserveWhitespace** is set to **true** . The **XmlWriter**  
 3        to which you want to save.

4        WriteContentTo

5  
 6        [C#] public override void WriteContentTo(XmlWriter w);  
 7        [C++] public: void WriteContentTo(XmlWriter\* w);  
 8        [VB] Overrides Public Sub WriteContentTo(ByVal w As XmlWriter)  
 9        [JScript] public override function WriteContentTo(w : XmlWriter);

10  
 11        *Description*

12        Saves all the children of the **XmlDocument** node to the specified  
 13        **System.Xml.XmlWriter** .  
 14        This method is functionally equivalent to the  
 15        **System.Xml.XmlDocument.InnerXml** property. The **XmlWriter** to which you  
 16        want to save.

17        WriteTo

18  
 19        [C#] public override void WriteTo(XmlWriter w);  
 20        [C++] public: void WriteTo(XmlWriter\* w);  
 21        [VB] Overrides Public Sub WriteTo(ByVal w As XmlWriter)  
 22        [JScript] public override function WriteTo(w : XmlWriter);

23  
 24        *Description*

25        Saves the **XmlDocument** node to the specified **System.Xml.XmlWriter** .

This method is functionally equivalent to the **System.Xml.XmlNode.OuterXml** property. The **XmlWriter** to which you want to save.

XmlDocumentFragment class (System.Xml)

WriteTo

### *Description*

Represents a lightweight object that is useful for tree insert operations.

XmlDocumentFragment

### *Example Syntax:*

WriteTo

[C#] protected internal XmlDocumentFragment(XmlDocument ownerDocument);

[C++] internal: XmlDocumentFragment(XmlDocument\* ownerDocument);

[VB] Protected Friend Sub New(ByVal ownerDocument As XmlDocument)

[JScript] package function XmlDocumentFragment(ownerDocument :  
XmlDocument);

### *Description*

Attributes

BaseURI

ChildNodes

FirstChild

HasChildNodes

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25

InnerText

InnerXml

WriteTo

*Description*

Gets or sets the markup representing the children of this node.

Setting this property replaces the children of the node with the parsed contents of the given string. The parsing is done in the current namespace context.

IsReadOnly

Item

Item

LastChild

LocalName

WriteTo

*Description*

Gets the local name of the node.

Name

WriteTo

[C#] public override string Name {get;}

[C++] public: \_\_property virtual String\* get\_Name();

[VB] Overrides Public ReadOnly Property Name As String

1 [JScript] public function get Name() : String;

3 *Description*

4 Gets the qualified name of the node.

5 NamespaceURI

6 NextSibling

7 NodeType

8 WriteTo

11 *Description*

12 Gets the type of the current node.

13 OuterXml

14 OwnerDocument

15 WriteTo

18 *Description*

19 Gets the **System.Xml.XmlDocument** that contains this node.

20 When adding nodes to the current node, use the **XmlDocument** returned by  
21 the **OwnerDocument** property to create the node.

22 ParentNode

23 WriteTo

25 [C#] public override XmlNode ParentNode {get;}

1 [C++] public: \_\_property virtual XmlNode\* get\_ParentNode();

2 [VB] Overrides Public ReadOnly Property ParentNode As XmlNode

3 [JScript] public function get ParentNode() : XmlNode;

4  
5 *Description*

6 Gets the parent of this node (for nodes that can have parents).

7 Prefix

8 PreviousSibling

9 Value

10 CloneNode

11  
12 [C#] public override XmlNode CloneNode(bool deep);

13 [C++] public: XmlNode\* CloneNode(bool deep);

14 [VB] Overrides Public Function CloneNode(ByVal deep As Boolean) As  
15 XmlNode

16 [JScript] public override function CloneNode(deep : Boolean) : XmlNode;

17  
18 *Description*

19 Creates a duplicate of this node.

20 *Return Value:* The cloned node.

21 **CloneNode** serves as a copy constructor for nodes. To see how this method  
22 behaves with other node types, see  
23 **System.Xml.XmlNode.CloneNode(System.Boolean)** method in the **XmlNode**  
24 class. **true** to recursively clone the subtree under the specified node; **false** to clone  
25 only the node itself.

## WriteContentTo

[C#] public override void WriteContentTo(XmlWriter w);  
[C++] public: void WriteContentTo(XmlWriter\* w);  
[VB] Overrides Public Sub WriteContentTo(ByVal w As XmlWriter)  
[JScript] public override function WriteContentTo(w : XmlWriter);

### *Description*

Saves all the children of the node to the specified XmlWriter. The XmlWriter where you want to save the current node.

## WriteTo

[C#] public override void WriteTo(XmlWriter w);  
[C++] public: void WriteTo(XmlWriter\* w);  
[VB] Overrides Public Sub WriteTo(ByVal w As XmlWriter)  
[JScript] public override function WriteTo(w : XmlWriter);

### *Description*

Saves the node to the specified XmlWriter. The XmlWriter where you want to save the node.

XmlDocumentType class (System.Xml)

## WriteTo

### *Description*

Represents the document type declaration.

XmlDocumentType

*Example Syntax:*

WriteTo

[C#] protected internal XmlDocumentType(string name, string publicId, string  
systemId, string internalSubset, XmlDocument doc);

[C++] internal: XmlDocumentType(String\* name, String\* publicId, String\*  
systemId, String\* internalSubset, XmlDocument\* doc);

[VB] Protected Friend Sub New(ByVal name As String, ByVal publicId As  
String, ByVal systemId As String, ByVal internalSubset As String, ByVal doc As  
XmlDocument)

[JScript] package function XmlDocumentType(name : String, publicId : String,  
systemId : String, internalSubset : String, doc : XmlDocument);

*Description*

Attributes

BaseURI

ChildNodes

Entities

WriteTo

*Description*

1 Gets the collection of **System.Xml.XmlEntity** nodes declared in the  
2 document type declaration.

3 FirstChild

4 HasChildNodes

5 InnerText

6 InnerXml

7 InternalSubset

8 WriteTo

9  
10  
11 *Description*

12 Gets the value of the DTD internal subset on the DOCTYPE declaration.

13 IsReadOnly

14 WriteTo

15  
16 [C#] public override bool IsReadOnly {get;}

17 [C++] public: \_\_property virtual bool get\_IsReadOnly();

18 [VB] Overrides Public ReadOnly Property IsReadOnly As Boolean

19 [JScript] public function get IsReadOnly() : Boolean;

20  
21 *Description*

22 Gets a value indicating whether the node is read-only.

23 A read-only node is one whose properties, attributes, or children cannot be  
24 changed. However, you can remove a read-only node from the tree and insert it  
25 somewhere else.



MSI-865US.APP

1	Item
2	Item
3	LastChild
4	LocalName
5	WriteTo
6	
7	
8	<i>Description</i>
9	Gets the local name of the node.
10	Name
11	WriteTo
12	
13	[C#] public override string Name {get;}
14	[C++] public: __property virtual String* get_Name();
15	[VB] Overrides Public ReadOnly Property Name As String
16	[JScript] public function get Name() : String;
17	
18	<i>Description</i>
19	Gets the qualified name of the node.
20	NamespaceURI
21	NextSibling
22	NodeType
23	WriteTo
24	
25	

1  
2  
3 *Description*

4 Gets the type of the current node.

5 Notations

6 WriteTo

7  
8 [C#] public XmlNamedNodeMap Notations {get;}

9 [C++] public: \_\_property XmlNamedNodeMap\* get\_Notations();

10 [VB] Public ReadOnly Property Notations As XmlNamedNodeMap

11 [JScript] public function get Notations() : XmlNamedNodeMap;

12  
13 *Description*

14 Gets the collection of **System.Xml.XmlNotation** nodes present in the  
15 document type declaration.

16 OuterXml

17 OwnerDocument

18 ParentNode

19 Prefix

20 PreviousSibling

21 PublicId

22 WriteTo

23  
24  
25 *Description*

Gets the value of the public identifier on the DOCTYPE declaration.

SystemId

WriteTo

[C#] public string SystemId {get;}

[C++] public: \_\_property String\* get\_SystemId();

[VB] Public ReadOnly Property SystemId As String

[JScript] public function get SystemId() : String;

### Description

Gets the value of the system identifier on the DOCTYPE declaration.

Value

CloneNode

[C#] public override XmlNode CloneNode(bool deep);

[C++] public: XmlNode\* CloneNode(bool deep);

[VB] Overrides Public Function CloneNode(ByVal deep As Boolean) As

XmlNode

[JScript] public override function CloneNode(deep : Boolean) : XmlNode;

### Description

Creates a duplicate of this node.

*Return Value:* The duplicate node.

This method serves as a generic copy constructor for nodes. The duplicate node has no parent (ParentNode returns null). true to recursively clone the subtree

1 under the specified node; false to clone only the node itself (and its attributes if the  
2 node is an XmlElement).

### 3 WriteContentTo

4  
5 [C#] public override void WriteContentTo(XmlWriter w);

6 [C++] public: void WriteContentTo(XmlWriter\* w);

7 [VB] Overrides Public Sub WriteContentTo(ByVal w As XmlWriter)

8 [JScript] public override function WriteContentTo(w : XmlWriter);

### 9 10 *Description*

11 Saves all the children of the node to the specified XmlWriter. The  
12 XmlWriter where you want to save the current node.

### 13 WriteTo

14  
15 [C#] public override void WriteTo(XmlWriter w);

16 [C++] public: void WriteTo(XmlWriter\* w);

17 [VB] Overrides Public Sub WriteTo(ByVal w As XmlWriter)

18 [JScript] public override function WriteTo(w : XmlWriter);

### 19 20 *Description*

21 Saves the node to the specified XmlWriter. The XmlWriter where you want  
22 to save the node.

23 XmlElement class (System.Xml)

### 24 WriteTo

*Description*

Represents an element.

XmlElement

*Example Syntax:*

WriteTo

[C#] protected internal XmlElement(string prefix, string localName, string namespaceURI, XmlDocument doc);

[C++] internal: XmlElement(String\* prefix, String\* localName, String\* namespaceURI, XmlDocument\* doc);

[VB] Protected Friend Sub New(ByVal prefix As String, ByVal localName As String, ByVal namespaceURI As String, ByVal doc As XmlDocument)

[JScript] package function XmlElement(prefix : String, localName : String, namespaceURI : String, doc : XmlDocument);

*Description*

Attributes

WriteTo

[C#] public override XmlAttributeCollection Attributes {get;}

[C++] public: \_\_property virtual XmlAttributeCollection\* get\_Attributes();

[VB] Overrides Public ReadOnly Property Attributes As XmlAttributeCollection

[JScript] public function get Attributes() : XmlAttributeCollection;

1  
2 *Description*

3 Gets an **System.Xml.XmlAttributeCollection** containing the list of  
4 attributes for this node.

5 BaseURI

6 ChildNodes

7 FirstChild

8 HasAttributes

9 WriteTo

10  
11  
12 *Description*

13 Gets a value indicating whether the current node has any attributes.

14 HasChildNodes

15 InnerText

16 WriteTo

17  
18  
19 *Description*

20 Gets or sets the concatenated values of the node and all its children.

21 Setting this property replaces all the children with the parsed contents of the  
22 given string.

23 InnerXml

24 WriteTo

```

1
2 [C#] public override string InnerXml {get; set;}
3 [C++] public: __property virtual String* get_InnerXml();public: __property
4 virtual void set_InnerXml(String*);
5 [VB] Overrides Public Property InnerXml As String
6 [JScript] public function get InnerXml() : String;public function set
7 InnerXml(String);
8

```

### *Description*

Gets or sets the markup representing just the children of this node.

Setting this property replaces the children of the node with the parsed contents of the given string. The parsing is done in the current namespace context.

IsEmpty

WriteTo

```

15
16 [C#] public bool IsEmpty {get; set;}
17 [C++] public: __property bool get_IsEmpty();public: __property void
18 set_IsEmpty(bool);
19 [VB] Public Property IsEmpty As Boolean
20 [JScript] public function get IsEmpty() : Boolean;public function set
21 IsEmpty(Boolean);
22

```

### *Description*

Gets or sets the tag format of the element.

IsReadOnly

Copyright © 2000 Microsoft Corporation. All rights reserved.

1 Item  
2 Item  
3 LastChild  
4 LocalName  
5 WriteTo

8 *Description*

9 Gets the local name of the current node.  
10 If the node does not have a prefix, **LocalName** is the same as

11 **System.Xml.XmlElement.Name** .

12 Name  
13 WriteTo

14  
15 [C#] public override string Name {get;}  
16 [C++] public: \_\_property virtual String\* get\_Name();  
17 [VB] Overrides Public ReadOnly Property Name As String  
18 [JScript] public function get Name() : String;

19  
20 *Description*

21 Gets the qualified name of the node.  
22 NamespaceURI  
23 WriteTo

24  
25 [C#] public override string NamespaceURI {get;}



1 [C++] public: \_\_property virtual String\* get\_NamespaceURI();

2 [VB] Overrides Public ReadOnly Property NamespaceURI As String

3 [JScript] public function get NamespaceURI() : String;

4  
5 *Description*

6 Gets the namespace URI of this node.

7 This is the namespace URI specified at creation time. For example,

8 **NamespaceURI** is urn:samples for the element The following example displays  
9 information on the ISBN element.

10 NextSibling

11 WriteTo

12  
13 [C#] public override XmlNode NextSibling {get;}

14 [C++] public: \_\_property virtual XmlNode\* get\_NextSibling();

15 [VB] Overrides Public ReadOnly Property NextSibling As XmlNode

16 [JScript] public function get NextSibling() : XmlNode;

17  
18 *Description*

19 Gets the **System.Xml.XmlNode** immediately following this element.

20 NodeType

21 WriteTo

22  
23 [C#] public override XmlNodeType NodeType {get;}

24 [C++] public: \_\_property virtual XmlNodeType get\_NodeType();

25 [VB] Overrides Public ReadOnly Property NodeType As XmlNodeType

1 [JScript] public function get NodeType() : XmlNodeType;

2  
3 *Description*

4 Gets the type of the current node.

5 OuterXml

6 OwnerDocument

7 WriteTo

8  
9  
10 *Description*

11 Gets the **System.Xml.XmlDocument** to which this node belongs.

12 When adding nodes to the current node, use the **XmlDocument** returned by  
13 the **OwnerDocument** property to create the node.

14 ParentNode

15 Prefix

16 WriteTo

17  
18  
19 *Description*

20 Gets or sets the namespace prefix of this node.

21 Setting this property changes the **System.Xml.XmlElement.Name**  
22 property, which holds the qualified name for an **XmlElement** .

23 PreviousSibling

24 Value

25 CloneNode

[C#] public override XmlNode CloneNode(bool deep);

[C++] public: XmlNode\* CloneNode(bool deep);

[VB] Overrides Public Function CloneNode(ByVal deep As Boolean) As XmlNode

[JScript] public override function CloneNode(deep : Boolean) : XmlNode;

### *Description*

Creates a duplicate of this node.

*Return Value:* The cloned node.

This method serves as a copy constructor for nodes. The duplicate node has no parent ( **System.Xml.XmlNode.ParentNode** returns **null** ). **true** to recursively clone the subtree under the specified node; **false** to clone only the node itself (and its attributes if the node is an **XmlElement** ).

### *GetAttribute*

[C#] public virtual string GetAttribute(string name);

[C++] public: virtual String\* GetAttribute(String\* name);

[VB] Overridable Public Function GetAttribute(ByVal name As String) As String

[JScript] public function GetAttribute(name : String) : String; Returns the attribute value for the specified attribute.

### *Description*

Returns the value for the attribute with the specified name.

*Return Value:* The value of the specified **System.Xml.XmlAttribute** as a string.



1 [JScript] public function GetAttributeNode(name : String) : XmlAttribute; Return  
2 the specified **System.Xml.XmlAttribute** .

3  
4 *Description*

5 Returns the **XmlAttribute** with the specified name.

6 *Return Value:* The specified **XmlAttribute** or **null** if a matching attribute was not  
7 found. The name of the attribute to retrieve. This is a qualified name. It is matched  
8 against the **Name** property of the matching node.

9 GetAttributeNode

10  
11 [C#] public virtual XmlAttribute GetAttributeNode(string localName, string  
12 namespaceURI);

13 [C++] public: virtual XmlAttribute\* GetAttributeNode(String\* localName,  
14 String\* namespaceURI);

15 [VB] Overridable Public Function GetAttributeNode(ByVal localName As String,  
16 ByVal namespaceURI As String) As XmlAttribute

17 [JScript] public function GetAttributeNode(localName : String, namespaceURI :  
18 String) : XmlAttribute;

19  
20 *Description*

21 Returns the **System.Xml.XmlAttribute** with the specified local name and  
22 namespace URI.

23 *Return Value:* The specified **XmlAttribute** or **null** if a matching attribute was not  
24 found. The local name of the attribute. The namespace URI of the attribute.

25 GetElementsByTagName

```

1
2 [C#] public virtual XmlNodeList GetElementsByTagName(string name);
3 [C++] public: virtual XmlNodeList* GetElementsByTagName(String* name);
4 [VB] Overridable Public Function GetElementsByTagName(ByVal name As
5 String) As XmlNodeList
6 [JScript] public function GetElementsByTagName(name : String) : XmlNodeList;
7 Returns an System.Xml.XmlNodeList containing a list of all descendant elements
8 that match the specified name.

```

### *Description*

Returns an **System.Xml.XmlNodeList** containing a list of all descendant elements that match the specified **System.Xml.XmlElement.Name**.

*Return Value:* An **System.Xml.XmlNodeList** containing a list of all matching nodes.

The nodes are placed in the order in which they would be encountered in a preorder traversal of the **System.Xml.XmlElement** tree. The name tag to match. This is a qualified name. It is matched against the **Name** property of the matching node. The asterik (\*) is a special value that matches all tags.

### **GetElementsByTagName**

```

21 [C#] public virtual XmlNodeList GetElementsByTagName(string localName,
22 string namespaceURI);
23 [C++] public: virtual XmlNodeList* GetElementsByTagName(String*
24 localName, String* namespaceURI);
25 [VB] Overridable Public Function GetElementsByTagName(ByVal localName As

```

String, ByVal namespaceURI As String) As XmlNodeList

[JScript] public function GetElementsByTagName(localName : String,  
namespaceURI : String) : XmlNodeList;

### *Description*

Returns an **System.Xml.XmlNodeList** containing a list of all descendant elements that match the specified **System.Xml.XmlElement.LocalName** and **System.Xml.XmlElement.NamespaceURI**.

*Return Value:* An **System.Xml.XmlNodeList** containing a list of all matching nodes.

The nodes are placed in the order in which they would be encountered in a preorder traversal of the **XmlElement** tree. The local name to match. The asterik (\*) is a special value that matches all tags. The namespace URI to match.

### **HasAttribute**

[C#] public virtual bool HasAttribute(string name);

[C++] public: virtual bool HasAttribute(String\* name);

[VB] Overridable Public Function HasAttribute(ByVal name As String) As  
Boolean

[JScript] public function HasAttribute(name : String) : Boolean; Determines  
whether the current node has the specified attribute.

### *Description*

Determines whether the current node has the specified attribute.

*Return Value:* **true** if the current node has the specified attribute; otherwise, **false**.

1 The name of the attribute to find. This is a qualified name. It is matched against  
2 the **Name** property of the matching node.

### 3 HasAttribute

4  
5 [C#] public virtual bool HasAttribute(string localName, string namespaceURI);

6 [C++] public: virtual bool HasAttribute(String\* localName, String\*  
7 namespaceURI);

8 [VB] Overridable Public Function HasAttribute(ByVal localName As String,  
9 ByVal namespaceURI As String) As Boolean

10 [JScript] public function HasAttribute(localName : String, namespaceURI :  
11 String) : Boolean;

### 12 13 *Description*

14 Determines whether the current node has an attribute with the specified  
15 local name and namespace URI.

16 *Return Value:* **true** if the current node has the specified attribute; otherwise, **false** .

17 The local name of the attribute to find. The namespace URI of the attribute to find.

### 18 RemoveAll

19  
20 [C#] public override void RemoveAll();

21 [C++] public: void RemoveAll();

22 [VB] Overrides Public Sub RemoveAll()

23 [JScript] public override function RemoveAll();

### 24 25 *Description*



Removes all the attributes and children of the current node.

## RemoveAllAttributes

[C#] public virtual void RemoveAllAttributes();

[C++] public: virtual void RemoveAllAttributes();

[VB] Overridable Public Sub RemoveAllAttributes()

[JScript] public function RemoveAllAttributes();

### *Description*

Removes all attributes from the element.

## RemoveAttribute

[C#] public virtual void RemoveAttribute(string name);

[C++] public: virtual void RemoveAttribute(String\* name);

[VB] Overridable Public Sub RemoveAttribute(ByVal name As String)

[JScript] public function RemoveAttribute(name : String); Removes the specified attribute.

### *Description*

Removes an attribute by name.

If the removed attribute is known to have a default value, an attribute immediately appears containing the default value and, if applicable, the corresponding namespace URI, local name, and prefix. The name of the attribute to remove. This is a qualified name. It is matched against the **Name** property of the matching node.

## RemoveAttribute

```
[C#] public virtual void RemoveAttribute(string localName, string
namespaceURI);
[C++] public: virtual void RemoveAttribute(String* localName, String*
namespaceURI);
[VB] Overridable Public Sub RemoveAttribute(ByVal localName As String,
ByVal namespaceURI As String)
[JScript] public function RemoveAttribute(localName : String, namespaceURI :
String);
```

### *Description*

Removes an attribute with the specified local name and namespace URI.

If the removed attribute is known to have a default value, an attribute immediately appears containing the default value and, if applicable, the corresponding namespace URI, local name, and prefix. The local name of the attribute to remove. The namespace URI of the attribute to remove.

## RemoveAttributeAt

```
[C#] public virtual XmlNode RemoveAttributeAt(int i);
[C++] public: virtual XmlNode* RemoveAttributeAt(int i);
[VB] Overridable Public Function RemoveAttributeAt(ByVal i As Integer) As
XmlNode
[JScript] public function RemoveAttributeAt(i : int) : XmlNode;
```

1  
2 *Description*

3 Removes the attribute node with the specified index from the element. The  
4 index of the node to remove. The first node has index 0.

5 RemoveAttributeNode

6  
7 [C#] public virtual XmlAttribute RemoveAttributeNode(XmlAttribute oldAttr);

8 [C++] public: virtual XmlAttribute\* RemoveAttributeNode(XmlAttribute\*  
9 oldAttr);

10 [VB] Overridable Public Function RemoveAttributeNode(ByVal oldAttr As  
11 XmlAttribute) As XmlAttribute

12 [JScript] public function RemoveAttributeNode(oldAttr : XmlAttribute) :

13 XmlAttribute; Removes an **System.Xml.XmlAttribute** .

14  
15 *Description*

16 Removes the specified **System.Xml.XmlAttribute** .

17 *Return Value:* The removed **XmlAttribute** or **null** if *oldAttr* is not an attribute  
18 node of the **XmlElement** . The **XmlAttribute** node to remove. If the removed  
19 attribute has a default value, it is immediately replaced.

20 RemoveAttributeNode

21  
22 [C#] public virtual XmlAttribute RemoveAttributeNode(string localName, string  
23 namespaceURI);

24 [C++] public: virtual XmlAttribute\* RemoveAttributeNode(String\* localName,  
25 String\* namespaceURI);

```

1 [VB] Overridable Public Function RemoveAttributeNode(ByVal localName As
2 String, ByVal namespaceURI As String) As XmlAttribute
3 [JScript] public function RemoveAttributeNode(localName : String,
4 namespaceURI : String) : XmlAttribute;
5

```

### Description

Removes the **System.Xml.XmlAttribute** specified by the local name and namespace URI.

*Return Value:* The removed **XmlAttribute** or **null** if the **XmlElement** does not have a matching attribute node. The local name of the attribute. The namespace URI of the attribute.

### SetAttribute

```

12
13
14 [C#] public virtual void SetAttribute(string name, string value);
15 [C++] public: virtual void SetAttribute(String* name, String* value);
16 [VB] Overridable Public Sub SetAttribute(ByVal name As String, ByVal value As
17 String)
18 [JScript] public function SetAttribute(name : String, value : String); Sets the value
19 of the specified attribute.
20

```

### Description

Sets the value of the attribute with the specified name.

If an attribute with the same name is already present in the element, its value is changed to that of *value* . The name of the attribute to create or alter. This

is a qualified name. If the name contains a colon it is parsed into prefix and local name components. The value to set for the attribute.

### SetAttribute

```
[C#] public virtual string SetAttribute(string localName, string namespaceURI,  
string value);  
[C++] public: virtual String* SetAttribute(String* localName, String*  
namespaceURI, String* value);  
[VB] Overridable Public Function SetAttribute(ByVal localName As String,  
ByVal namespaceURI As String, ByVal value As String) As String  
[JScript] public function SetAttribute(localName : String, namespaceURI : String,  
value : String) : String;
```

### Description

Sets the value of the attribute with the specified local name and namespace URI. The local name of the attribute. The namespace URI of the attribute. The value to set for the attribute.

### SetAttributeNode

```
[C#] public virtual XmlAttribute SetAttributeNode(XmlAttribute newAttr);  
[C++] public: virtual XmlAttribute* SetAttributeNode(XmlAttribute* newAttr);  
[VB] Overridable Public Function SetAttributeNode(ByVal newAttr As  
XmlAttribute) As XmlAttribute  
[JScript] public function SetAttributeNode(newAttr : XmlAttribute) :  
XmlAttribute; Adds a new System.Xml.XmlAttribute .
```

## Description

Adds the specified **System.Xml.XmlAttribute** .

*Return Value:* If the attribute replaces an existing attribute with the same name, the old **XmlAttribute** is returned; otherwise, **null** is returned.

If an attribute with that name is already present in the element, it is replaced by the new one. The **XmlAttribute** node to add to the attribute collection for this element.

## SetAttributeNode

[C#] public virtual XmlAttribute SetAttributeNode(string localName, string namespaceURI);

[C++] public: virtual XmlAttribute\* SetAttributeNode(String\* localName, String\* namespaceURI);

[VB] Overridable Public Function SetAttributeNode(ByVal localName As String, ByVal namespaceURI As String) As XmlAttribute

[JScript] public function SetAttributeNode(localName : String, namespaceURI : String) : XmlAttribute;

## Description

Adds the specified **System.Xml.XmlAttribute** .

*Return Value:* The **XmlAttribute** to add.

The **XmlAttribute** does not have any children. Use **System.Xml.XmlAttribute.Value** to assign a text value to the attribute or use **System.Xml.XmlNode.AppendChild(System.Xml.XmlNode)** (or a similar

method) to add children to the attribute. The local name of the attribute. The namespace URI of the attribute.

### WriteContentTo

[C#] public override void WriteContentTo(XmlWriter w);

[C++] public: void WriteContentTo(XmlWriter\* w);

[VB] Overrides Public Sub WriteContentTo(ByVal w As XmlWriter)

[JScript] public override function WriteContentTo(w : XmlWriter);

### Description

Saves all the children of the node to the specified **System.Xml.XmlWriter**. The **XmlWriter** to which you want to save.

### WriteTo

[C#] public override void WriteTo(XmlWriter w);

[C++] public: void WriteTo(XmlWriter\* w);

[VB] Overrides Public Sub WriteTo(ByVal w As XmlWriter)

[JScript] public override function WriteTo(w : XmlWriter);

### Description

Saves the current node to the specified **System.Xml.XmlWriter**. The **XmlWriter** to which you want to save.

XmlEntity class (System.Xml)

### WriteTo

*Description*

Represents an entity declaration: .

Attributes

BaseURI

WriteTo

*Description*

Gets the base URI of the current node.

A networked XML document is comprised of chunks of data aggregated using various W3C standard inclusion mechanisms and therefore contains nodes that come from different places. The **BaseURI** tells you where these nodes came from.

ChildNodes

FirstChild

HasChildNodes

InnerText

WriteTo

*Description*

Gets the concatenated values of the entity node and all its children.

**XmlEntity** nodes are read-only. Setting this property throws an exception.



InnerXml

WriteTo

[C#] public override string InnerXml {get; set;}

[C++] public: \_\_property virtual String\* get\_InnerXml();public: \_\_property  
virtual void set\_InnerXml(String\*);

[VB] Overrides Public Property InnerXml As String

[JScript] public function get InnerXml() : String;public function set  
InnerXml(String);

### *Description*

Gets the markup representing the children of this node.

**XmlEntity** nodes are read-only. Setting this property throws an exception.

IsReadOnly

WriteTo

[C#] public override bool IsReadOnly {get;}

[C++] public: \_\_property virtual bool get\_IsReadOnly();

[VB] Overrides Public ReadOnly Property IsReadOnly As Boolean

[JScript] public function get IsReadOnly() : Boolean;

### *Description*

Gets a value indicating whether the node is read-only.

A read-only node is one whose properties, attributes, or children cannot be changed. You can remove a read-only node from the tree and insert it somewhere else.

Item

Item

LastChild

LocalName

WriteTo

#### *Description*

Gets the name of the node without the namespace prefix.

Name

WriteTo

[C#] public override string Name {get;}

[C++] public: \_\_property virtual String\* get\_Name();

[VB] Overrides Public ReadOnly Property Name As String

[JScript] public function get Name() : String;

#### *Description*

Gets the name of the node.

NamespaceURI

NextSibling

NodeType

WriteTo

*Description*

Gets the type of the node.

NotationName

WriteTo

[C#] public string NotationName {get;}

[C++] public: \_\_property String\* get\_NotationName();

[VB] Public ReadOnly Property NotationName As String

[JScript] public function get NotationName() : String;

*Description*

Gets the name of the optional NDATA attribute on the entity declaration.

OuterXml

WriteTo

[C#] public override string OuterXml {get;}

[C++] public: \_\_property virtual String\* get\_OuterXml();

[VB] Overrides Public ReadOnly Property OuterXml As String

[JScript] public function get OuterXml() : String;

*Description*

Gets the markup representing this node and all its children.

1 OwnerDocument  
 2 ParentNode  
 3 Prefix  
 4 PreviousSibling  
 5 PublicId  
 6 WriteTo

9 *Description*

10 Gets the value of the public identifier on the entity declaration.

11 SystemId  
 12 WriteTo

13  
 14 [C#] public string SystemId {get;}  
 15 [C++] public: \_\_property String\* get\_SystemId();  
 16 [VB] Public ReadOnly Property SystemId As String  
 17 [JScript] public function get SystemId() : String;

19 *Description*

20 Gets the value of the system identifier on the entity declaration.

21 Value  
 22 CloneNode

23  
 24 [C#] public override XmlNode CloneNode(bool deep);  
 25 [C++] public: XmlNode\* CloneNode(bool deep);

[VB] Overrides Public Function CloneNode(ByVal deep As Boolean) As

XmlNode

[JScript] public override function CloneNode(deep : Boolean) : XmlNode;

### Description

Creates a duplicate of this node. Entity nodes cannot be cloned. Calling this method on an **XmlNode** object throws an exception. **true** to recursively clone the subtree under the specified node; **false** to clone only the node itself.

### WriteContentTo

[C#] public override void WriteContentTo(XmlWriter w);

[C++] public: void WriteContentTo(XmlWriter\* w);

[VB] Overrides Public Sub WriteContentTo(ByVal w As XmlWriter)

[JScript] public override function WriteContentTo(w : XmlWriter);

### Description

Saves all the children of the node to the specified **System.Xml.XmlWriter**. The **XmlWriter** to which you want to save.

### WriteTo

[C#] public override void WriteTo(XmlWriter w);

[C++] public: void WriteTo(XmlWriter\* w);

[VB] Overrides Public Sub WriteTo(ByVal w As XmlWriter)

[JScript] public override function WriteTo(w : XmlWriter);

1  
2 *Description*

3 Saves the node to the specified **System.Xml.XmlWriter** . The **XmlWriter**  
4 to which you want to save.

5 XmlEntityReference class (System.Xml)

6 WriteTo  
7  
8

9 *Description*

10 Represents an entity reference node.

11 XmlEntityReference

12 *Example Syntax:*

13 WriteTo  
14

15 [C#] protected internal XmlEntityReference(string name, XmlDocument doc);

16 [C++] internal: XmlEntityReference(String\* name, XmlDocument\* doc);

17 [VB] Protected Friend Sub New(ByVal name As String, ByVal doc As  
18 XmlDocument)

19 [JScript] package function XmlEntityReference(name : String, doc :  
20 XmlDocument);  
21

22 *Description*

23 Attributes

24 BaseURI

25 WriteTo

### Description

Gets the base URI of the current node.

A networked XML document is comprised of chunks of data aggregated using various W3C standard inclusion mechanisms and therefore contains nodes that come from different places. The **BaseURI** tells you where these nodes came from. If there is no base URI for the nodes being returned (maybe they were parsed from an in-memory string), String.Empty is returned.

ChildNodes

FirstChild

HasChildNodes

InnerText

InnerXml

IsReadOnly

WriteTo

### Description

Gets a value indicating whether the node is read-only.

A read-only node is one whose properties, attributes, or children cannot be changed. However, you can remove a read-only node from the tree and insert it somewhere else.

Item

Item

1 LastChild

2 LocalName

3 WriteTo

4

5

6 *Description*

7 Gets the local name of the node.

8 Name

9 WriteTo

10

11 [C#] public override string Name {get;}

12 [C++] public: \_\_property virtual String\* get\_Name();

13 [VB] Overrides Public ReadOnly Property Name As String

14 [JScript] public function get Name() : String;

15

16 *Description*

17 Gets the name of the node.

18 NamespaceURI

19 NextSibling

20 NodeType

21 WriteTo

22

23

24 *Description*

25 Gets the type of the node.



OuterXml  
 OwnerDocument  
 ParentNode  
 Prefix  
 PreviousSibling  
 Value  
 WriteTo

### Description

Gets or sets the value of the node.

CloneNode

[C#] public override XmlNode CloneNode(bool deep);

[C++] public: XmlNode\* CloneNode(bool deep);

[VB] Overrides Public Function CloneNode(ByVal deep As Boolean) As

XmlNode

[JScript] public override function CloneNode(deep : Boolean) : XmlNode;

### Description

Creates a duplicate of this node.

*Return Value:* The cloned node.

This method serves as a copy constructor for nodes. The cloned node has no parent ( **System.Xml.XmlNode.ParentNode** returns **null** ). **true** to recursively clone the subtree under the specified node; **false** to clone only the node itself. For

**XmlEntityReference** nodes, this method always returns an entity reference node with no children. The replacement text is set when the node is inserted into a parent.

WriteContentTo

[C#] public override void WriteContentTo(XmlWriter w);

[C++] public: void WriteContentTo(XmlWriter\* w);

[VB] Overrides Public Sub WriteContentTo(ByVal w As XmlWriter)

[JScript] public override function WriteContentTo(w : XmlWriter);

#### *Description*

Saves all the children of the node to the specified XmlWriter. The XmlWriter where you want to save the current node.

WriteTo

[C#] public override void WriteTo(XmlWriter w);

[C++] public: void WriteTo(XmlWriter\* w);

[VB] Overrides Public Sub WriteTo(ByVal w As XmlWriter)

[JScript] public override function WriteTo(w : XmlWriter);

#### *Description*

Saves the node to the specified XmlWriter. The XmlWriter where you want to save the node.

XmlException class (System.Xml)

WriteTo

### Description

Returns detailed information about the last exception.

**XmlException**

*Example Syntax:*

**WriteTo**

[C#] public XmlException(SerializationInfo info, StreamingContext context);

[C++] public: XmlException(SerializationInfo\* info, StreamingContext context);

[VB] Public Sub New(ByVal info As SerializationInfo, ByVal context As StreamingContext)

[JScript] public function XmlException(info : SerializationInfo, context : StreamingContext); Initializes a new instance of the **XmlException** class.

### Description

Initializes a new instance of the **XmlException** class using the information in the **System.Runtime.Serialization.SerializationInfo** and **System.Runtime.Serialization.StreamingContext** objects. The **SerializationInfo** object containing all the properties of an **XmlException**. The **StreamingContext** object containing the context information.

**XmlException**

*Example Syntax:*

**WriteTo**

```

1
2 [C#] public XmlException(string message, Exception innerException);
3 [C++] public: XmlException(String* message, Exception* innerException);
4 [VB] Public Sub New(ByVal message As String, ByVal innerException As
5 Exception)
6 [JScript] public function XmlException(message : String, innerException :
7 Exception);
8

```

### *Description*

Initializes a new instance of the **XmlException** class. The description of the error condition. The **System.Exception** that threw the **XmlException**, if any. This value can be **null**.

HelpLink

HResult

InnerException

LineNumber

WriteTo

### *Description*

Gets the line number indicating where the error occurred.

LinePosition

WriteTo

```

25 [C#] public int LinePosition {get;}

```

[C++] public: \_\_property int get\_LinePosition();

[VB] Public ReadOnly Property LinePosition As Integer

[JScript] public function get LinePosition() : int;

### *Description*

Gets the line position indicating where the error occurred.

Message

WriteTo

[C#] public override string Message {get;}

[C++] public: \_\_property virtual String\* get\_Message();

[VB] Overrides Public ReadOnly Property Message As String

[JScript] public function get Message() : String;

### *Description*

Gets the error message describing the exception.

Source

StackTrace

TargetSite

GetObjectData

[C#] public override void GetObjectData(SerializationInfo info, StreamingContext context);

[C++] public: void GetObjectData(SerializationInfo\* info, StreamingContext context);

```

1 [VB] Overrides Public Sub GetObjectData(ByVal info As SerializationInfo,
2   ByVal context As StreamingContext)
3 [JScript] public override function GetObjectData(info : SerializationInfo, context :
4   StreamingContext);
5

```

### *Description*

Streams all the **XmlException** properties into the **System.Runtime.Serialization.SerializationInfo** class for the given **System.Runtime.Serialization.StreamingContext** . The **SerializationInfo** object. The **StreamingContext** object.

XmlImplementation class (System.Xml)

ToString

### *Description*

Defines the context for a set of **System.Xml.XmlDocument** objects. **XmlDocument** objects that are created from the same implementation share the same **System.Xml.XmlNameTable** . This enables users to compare attribute and element names between the objects more efficiently.

XmlImplementation

*Example Syntax:*

ToString

```

24 [C#] public XmlImplementation();
25 [C++] public: XmlImplementation();

```

1 [VB] Public Sub New()

2 [JScript] public function XmlImplementation();

3  
4 *Description*

5       Initializes a new instance of the XmlImplementation class.

6       CreateDocument

7  
8 [C#] public virtual XmlDocument CreateDocument();

9 [C++] public: virtual XmlDocument\* CreateDocument();

10 [VB] Overridable Public Function CreateDocument() As XmlDocument

11 [JScript] public function CreateDocument() : XmlDocument;

12  
13 *Description*

14       Creates a new **System.Xml.XmlDocument** .

15 *Return Value:* The new **XmlDocument** object.

16       **XmlDocument** objects created from the same implementation share the  
17 same name table. This enables users to compare attribute and element names  
18 between the objects more efficiently.

19       HasFeature

20  
21 [C#] public bool HasFeature(string strFeature, string strVersion);

22 [C++] public: bool HasFeature(String\* strFeature, String\* strVersion);

23 [VB] Public Function HasFeature(ByVal strFeature As String, ByVal strVersion  
24 As String) As Boolean

25 [JScript] public function HasFeature(strFeature : String, strVersion : String) :

Boolean;

### Description

Tests if the DOM implementation implements a specific feature.

**Return Value:** **true** if the feature is implemented in the specified version; otherwise, **false** . The package name of the feature to test. This name is not case-sensitive. This is the version number of the package name to test. If the version is not specified (**null**), supporting any version of the feature causes the method to return **true** .

XmlNode class (System.Xml)

ToString

### Description

Gets the node immediately preceding or following this node.

Attributes

BaseURI

ChildNodes

FirstChild

HasChildNodes

InnerText

InnerXml

IsReadOnly

Item

Item



- 1 LastChild
- 2 LocalName
- 3 Name
- 4 NamespaceURI
- 5 NextSibling
- 6 ToString

- 7
- 8
- 9 *Description*
- 10 Gets the node immediately following this node.
- 11 NodeType
- 12 OuterXml
- 13 OwnerDocument
- 14 ParentNode
- 15 Prefix
- 16 PreviousSibling
- 17 ToString

- 18
- 19
- 20 *Description*
- 21 Gets the node immediately preceding this node.
- 22 Value
- 23 XmlNamedNodeMap class (System.Xml)
- 24 WriteTo

25

1  
2  
3 *Description*

4 Represents a collection of nodes that can be accessed by name or index.

5 Count

6 WriteTo

7  
8 [C#] public virtual int Count {get;}

9 [C++] public: \_\_property virtual int get\_Count();

10 [VB] Overridable Public ReadOnly Property Count As Integer

11 [JScript] public function get Count() : int;

12  
13 *Description*

14 Gets the number of nodes in the **XmlNamedNodeMap** .

15 GetEnumerator

16  
17 [C#] public virtual IEnumerator GetEnumerator();

18 [C++] public: virtual IEnumerator\* GetEnumerator();

19 [VB] Overridable Public Function GetEnumerator() As IEnumerator

20 [JScript] public function GetEnumerator() : IEnumerator;

21  
22 *Description*

23 Provides support for the "foreach" style iteration over the collection of  
24 nodes in the **XmlNamedNodeMap** .

25 *Return Value:* An **System.Collections.IEnumerator** .

## GetNamedItem

```
[C#] public virtual XmlNode GetNamedItem(string name);  
[C++] public: virtual XmlNode* GetNamedItem(String* name);  
[VB] Overridable Public Function GetNamedItem(ByVal name As String) As  
XmlNode  
[JScript] public function GetNamedItem(name : String) : XmlNode; Retrieves the  
specified System.Xml.XmlNode from the collection of nodes in the  
System.Xml.XmlNamedNodeMap .
```

### *Description*

Retrieves an **System.Xml.XmlNode** specified by name.

*Return Value:* An **XmlNode** with the specified name or **null** if a matching node is not found. The qualified name of the node to retrieve. It is matched against the **System.Xml.XmlNode.Name** property of the matching node.

## GetNamedItem

```
[C#] public virtual XmlNode GetNamedItem(string localName, string  
namespaceURI);  
[C++] public: virtual XmlNode* GetNamedItem(String* localName, String*  
namespaceURI);  
[VB] Overridable Public Function GetNamedItem(ByVal localName As String,  
ByVal namespaceURI As String) As XmlNode  
[JScript] public function GetNamedItem(localName : String, namespaceURI :  
String) : XmlNode;
```

## Description

Retrieves a node with the matching **System.Xml.XmlNode.LocalName** and **System.Xml.XmlNode.NamespaceURI** .

*Return Value:* An **System.Xml.XmlNode** with the matching local name and namespace URI or **null** if a matching node was not found. The local name of the node to retrieve. The namespace URI of the node to retrieve.

## Item

[C#] public virtual XmlNode Item(int index);

[C++] public: virtual XmlNode\* Item(int index);

[VB] Overridable Public Function Item(ByVal index As Integer) As XmlNode

[JScript] public function Item(index : int) : XmlNode;

## Description

Retrieves the node at the specified index in the **XmlNamedNodeMap** .

*Return Value:* The **System.Xml.XmlNode** at the specified index. If *index* is greater than or equal to the **System.Xml.XmlNamedNodeMap.Count** property, **null** is returned. The index position of the node to retrieve from the **XmlNamedNodeMap** . The index is zero-based; therefore, the first node's index is 0 and the last node's index is **System.Xml.XmlNamedNodeMap.Count - 1**.

## RemoveNamedItem

[C#] public virtual XmlNode RemoveNamedItem(string name);

[C++] public: virtual XmlNode\* RemoveNamedItem(String\* name);

1 [VB] Overridable Public Function RemoveNamedItem(ByVal name As String) As  
2 XmlNode  
3 [JScript] public function RemoveNamedItem(name : String) : XmlNode; Removes  
4 the specified node from the **XmlNamedNodeMap** .

5  
6 *Description*

7 Removes the node from the **XmlNamedNodeMap** .

8 *Return Value:* The **XmlNode** removed from this **XmlNamedNodeMap** or **null** if  
9 a matching node was not found.

10 If the removed node is an **System.Xml.XmlAttribute** that contains a  
11 default value, it is immediately replaced in the **XmlNamedNodeMap** . The  
12 qualified name of the node to remove. The name is matched against the  
13 **System.Xml.XmlNode.Name** property of the matching node.

14 **RemoveNamedItem**

15  
16 [C#] public virtual XmlNode RemoveNamedItem(string localName, string  
17 namespaceURI);

18 [C++] public: virtual XmlNode\* RemoveNamedItem(String\* localName, String\*  
19 namespaceURI);

20 [VB] Overridable Public Function RemoveNamedItem(ByVal localName As  
21 String, ByVal namespaceURI As String) As XmlNode

22 [JScript] public function RemoveNamedItem(localName : String, namespaceURI :  
23 String) : XmlNode;

24  
25 *Description*

Removes a node with the matching **System.Xml.XmlNode.LocalName** and **System.Xml.XmlNode.NamespaceURI** .

*Return Value:* The **System.Xml.XmlNode** removed or **null** if a matching node was not found.

If the removed node is an **System.Xml.XmlAttribute** that contains a default value, it is immediately replaced in this **XmlNamedNodeMap** . The local name of the node to remove. The namespace URI of the node to remove.

SetNamedItem

[C#] public virtual XmlNode SetNamedItem(XmlNode node);

[C++] public: virtual XmlNode\* SetNamedItem(XmlNode\* node);

[VB] Overridable Public Function SetNamedItem(ByVal node As XmlNode) As XmlNode

[JScript] public function SetNamedItem(node : XmlNode) : XmlNode;

*Description*

Adds an **System.Xml.XmlNode** using its **System.Xml.XmlNode.Name** property

*Return Value:* If the *node* replaces an existing node with the same name, the old node is returned; otherwise, **null** is returned. An **XmlNode** to store in the **XmlNamedNodeMap** . If a node with that name is already present in the map, it is replaced by the new one.

XmlNamespaceManager class (System.Xml)

ToString

### Description

Resolves, adds and removes namespaces to a collection and provide scope management for these namespaces. This class is used by the **System.Xml.Xsl.XsltContext** and **System.Xml.XmlReader** classes.

**XmlNamespaceManager** stores prefixes and namespaces as strings.

XmlNamespaceManager

*Example Syntax:*

ToString

[C#] public XmlNamespaceManager(XmlNameTable nameTable);

[C++] public: XmlNamespaceManager(XmlNameTable\* nameTable);

[VB] Public Sub New(ByVal nameTable As XmlNameTable)

[JScript] public function XmlNamespaceManager(nameTable : XmlNameTable);

### Description

Initializes a new instance of the **XmlNamespaceManager** class with the specified **System.Xml.XmlNameTable** .

The name table is used to look up prefixes and namespaces. An existing name table with pre-atomized strings can be specified in the constructor. There are several advantages in doing so. If a XmlReader's name table is used, after each read any namespace and prefix strings pushed into the name table can be re-used by **XmlNamespaceManager** . Also, because **XmlNamespaceManager** is used

internally by the **XmlReader** , it can take advantage of the reader's ability to atomize strings. The **XmlNameTable** to use.

DefaultNamespace

ToString

[C#] public virtual string DefaultNamespace {get;}

[C++] public: \_\_property virtual String\* get\_DefaultNamespace();

[VB] Overridable Public ReadOnly Property DefaultNamespace As String

[JScript] public function get DefaultNamespace() : String;

#### *Description*

Gets the namespace URI for the default namespace.

This method is equivalent to calling LookupNamespace(String.Empty).

NameTable

ToString

[C#] public XmlNameTable NameTable {get;}

[C++] public: \_\_property XmlNameTable\* get\_NameTable();

[VB] Public ReadOnly Property NameTable As XmlNameTable

[JScript] public function get NameTable() : XmlNameTable;

#### *Description*

Gets the **System.Xml.XmlNameTable** associated with this object.

The name table is used to look up prefixes and namespace URIs.

AddNamespace



1  
2 [C#] public virtual void AddNamespace(string prefix, string uri);

3 [C++] public: virtual void AddNamespace(String\* prefix, String\* uri);

4 [VB] Overridable Public Sub AddNamespace(ByVal prefix As String, ByVal uri  
5 As String)

6 [JScript] public function AddNamespace(prefix : String, uri : String);

7  
8 *Description*

9 Adds the given namespace to the collection.

10 **XmlNamespaceManager** does not check *prefix* and *uri* for conformance.

11 The prefix to associate with the namespace being added. Use String.Empty to add  
12 a default namespace. The namespace to add.

13 GetEnumerator

14  
15 [C#] public virtual IEnumerator GetEnumerator();

16 [C++] public: virtual IEnumerator\* GetEnumerator();

17 [VB] Overridable Public Function GetEnumerator() As IEnumerator

18 [JScript] public function GetEnumerator() : IEnumerator;

19  
20 *Description*

21 Provides support for the "foreach" style iteration over the collection of  
22 namespaces in the **XmlNamespaceManager** .

23 *Return Value:* An **System.Collections.IEnumerator** .

24 HasNamespace

[C#] public virtual bool HasNamespace(string prefix);

[C++] public: virtual bool HasNamespace(String\* prefix);

[VB] Overridable Public Function HasNamespace(ByVal prefix As String) As Boolean

[JScript] public function HasNamespace(prefix : String) : Boolean;

### *Description*

Gets a value indicating whether the supplied prefix has a namespace defined for the current pushed scope.

*Return Value:* **true** if there is a namespace defined; otherwise, **false** .

To determine whether there is a default empty namespace defined, set *prefix* to String.Empty (or **null** ). If the method returns **true** , this indicates that xmlns= "". Returning **false** indicates that no default namespace is defined. Note that xmlns:x="" is illegal (the default namespace cannot be prefixed). The prefix of the namespace you want to find.

### *LookupNamespace*

[C#] public virtual string LookupNamespace(string prefix);

[C++] public: virtual String\* LookupNamespace(String\* prefix);

[VB] Overridable Public Function LookupNamespace(ByVal prefix As String) As String

[JScript] public function LookupNamespace(prefix : String) : String;

### *Description*

Gets the namespace URI for the specified prefix.

*Return Value:* Returns the namespace URI for *prefix* or **null** if there is no mapped namespace. The returned string is atomized. The prefix whose namespace URI you want to resolve. To match the default namespace, pass String.Empty.

### LookupPrefix

[C#] public virtual string LookupPrefix(string uri);

[C++] public: virtual String\* LookupPrefix(String\* uri);

[VB] Overridable Public Function LookupPrefix(ByVal uri As String) As String

[JScript] public function LookupPrefix(uri : String) : String;

### Description

Finds the prefix declared for the given namespace URI.

*Return Value:* The matching prefix. If there is no mapped prefix, the method returns String.Empty. If a null value is supplied then **null** is returned.

This method finds the mapped prefix by walking the stack (i.e. it looks globally). The supplied string must be atomized for the lookup to succeed. In other words the supplied string object must exist in the XmlNamespaceManager's **System.Xml.XmlNamespaceManager.NameTable** . The namespace to resolve for the prefix.

### PopScope

[C#] public virtual bool PopScope();

[C++] public: virtual bool PopScope();

[VB] Overridable Public Function PopScope() As Boolean

[JScript] public function PopScope() : Boolean;

### Description

Pops a namespace scope off the stack.

**Return Value:** **true** if there are namespace scopes left on the stack; **false** if there are no more namespaces to pop.

When you call this method, all of the namespaces which were added to **XmlNamespaceManager** (by calling **System.Xml.XmlNamespaceManager.AddNamespace(System.String, System.String)** ) since the last call to **PopScope** , are removed.

PushScope

[C#] public virtual void PushScope();

[C++] public: virtual void PushScope();

[VB] Overridable Public Sub PushScope()

[JScript] public function PushScope();

### Description

Pushes a namespace scope onto the stack.

After a call to this method all of the namespaces, which are added to **XmlNamespaceManager** (by calling **System.Xml.XmlNamespaceManager.AddNamespace(System.String, System.String)** ), belong to the pushed namespace scope.

RemoveNamespace

```

1
2 [C#] public virtual void RemoveNamespace(string prefix, string uri);
3 [C++] public: virtual void RemoveNamespace(String* prefix, String* uri);
4 [VB] Overridable Public Sub RemoveNamespace(ByVal prefix As String, ByVal
5 uri As String)
6 [JScript] public function RemoveNamespace(prefix : String, uri : String);
7

```

### *Description*

Removes the given namespace for the given prefix. The prefix for the namespace The namespace to remove for the given prefix. The namespace removed is from the current namespace scope. Namespaces outside the current scope are ignored. Exceptions are never thrown.

XmlNameTable class (System.Xml)

ToString

### *Description*

Table of atomized string objects.

Several classes, such as **System.Xml.XmlDocument** and **System.Xml.XmlReader** use the **XmlNameTable** class internally, to store attribute and element names. When an element or attribute name occurs multiple times in an XML document it is stored only once in the **XmlNameTable** . This enables you to do object comparisons on these strings rather than a more expensive string comparison.

XmlNameTable

*Example Syntax:*

**ToString**

[C#] protected XmlNameTable();

[C++] protected: XmlNameTable();

[VB] Protected Sub New()

[JScript] protected function XmlNameTable();

**Add**

[C#] public abstract string Add(string array);

[C++] public: virtual String\* Add(String\* array) = 0;

[VB] MustOverride Public Function Add(ByVal array As String) As String

[JScript] public abstract function Add(array : String) : String;

*Description*

When overridden in a derived class, atomizes the specified string and adds it to the **XmlNameTable** .

*Return Value:* The atomized string or the existing atom if one already exists. The name to add.

**Add**

[C#] public abstract string Add(char[] array, int offset, int length);

[C++] public: virtual String\* Add(\_\_wchar\_t array \_\_gc[], int offset, int length) = 0;

[VB] MustOverride Public Function Add(ByVal array() As Char, ByVal offset As

```
[JScript] public abstract function Add(array : Char[], offset : int, length : int) :
```

String; When overridden in a derived class, atomizes the specified string and adds it to the **XmlNameTable** .

### Description

When overridden in a derived class, atomizes the specified string and adds it to the **XmlNameTable**

*Return Value:* The atomized string or the existing atom if one already exists. If length is zero, String.Empty is returned. The character array containing the name to add. Zero based index into the array specifying the first character of the name. The number of characters in the name.

Get

```
[C#] public abstract string Get(string array);
```

```
[C++] public: virtual String* Get(String* array) = 0;
```

```
[VB] MustOverride Public Function Get(ByVal array As String) As String
```

```
[JScript] public abstract function Get(array : String) : String;
```

*Description*

When overridden in a derived class, gets the atomized **String** object containing the same value as the specified string.

*Return Value:* The atomized string or **null** if the string has not already been atomized. The name to look up.

Get

[C#] public abstract string Get(char[] array, int offset, int length);  
 [C++] public: virtual String\* Get(\_\_wchar\_t array \_\_gc[], int offset, int length) =  
 0;  
 [VB] MustOverride Public Function Get(ByVal array() As Char, ByVal offset As  
 Integer, ByVal length As Integer) As String  
 [JScript] public abstract function Get(array : Char[], offset : int, length : int) :  
 String; When overridden in a derived class, gets the atomized **String** object.

### *Description*

When overridden in a derived class, gets the atomized **String** object  
 containing the same characters as the specified range of characters in the given  
 array.

*Return Value:* The atomized string or **null** if the string has not already been  
 atomized. If *length* is zero, String.Empty is returned. The character array  
 containing the name to look up. The zero-based index into the array specifying the  
 first character of the name. The number of characters in the name.

XmlNode class (System.Xml)

ToString

### *Description*

Represents a single node in the XML document.

Attributes

ToString



1  
2 [C#] public virtual XmlAttributeCollection Attributes {get;}

3 [C++] public: \_\_property virtual XmlAttributeCollection\* get\_Attributes();

4 [VB] Overridable Public ReadOnly Property Attributes As

5 XmlAttributeCollection

6 [JScript] public function get Attributes() : XmlAttributeCollection;

7  
8 *Description*

9 Gets an **System.Xml.XmlAttributeCollection** containing the attributes of  
10 this node.

11 BaseURI

12 ToString

13  
14 [C#] public virtual string BaseURI {get;}

15 [C++] public: \_\_property virtual String\* get\_BaseURI();

16 [VB] Overridable Public ReadOnly Property BaseURI As String

17 [JScript] public function get BaseURI() : String;

18  
19 *Description*

20 Gets the base URI of the current node.

21 A networked XML document is comprised of chunks of data aggregated  
22 using various W3C standard inclusion mechanisms and therefore contains nodes  
23 that come from different places. The **BaseURI** tells you where these nodes came  
24 from.

25 ChildNodes

ToString

[C#] public virtual XmlNodeList ChildNodes {get;}

[C++] public: \_\_property virtual XmlNodeList\* get\_ChildNodes();

[VB] Overridable Public ReadOnly Property ChildNodes As XmlNodeList

[JScript] public function get ChildNodes() : XmlNodeList;

### Description

Gets all the children of the node.

FirstChild

ToString

[C#] public virtual XmlNode FirstChild {get;}

[C++] public: \_\_property virtual XmlNode\* get\_FirstChild();

[VB] Overridable Public ReadOnly Property FirstChild As XmlNode

[JScript] public function get FirstChild() : XmlNode;

### Description

Gets the first child of the node.

HasChildNodes

ToString

[C#] public virtual bool HasChildNodes {get;}

[C++] public: \_\_property virtual bool get\_HasChildNodes();

[VB] Overridable Public ReadOnly Property HasChildNodes As Boolean

[JScript] public function get HasChildNodes() : Boolean;

### Description

Gets a value indicating whether this node has any child nodes.

InnerText

ToString

[C#] public virtual string InnerText {get; set;}

[C++] public: \_\_property virtual String\* get\_InnerText();public: \_\_property

virtual void set\_InnerText(String\*);

[VB] Overridable Public Property InnerText As String

[JScript] public function get InnerText() : String;public function set

InnerText(String);

### Description

Gets or sets the concatenated values of the node and all its children.

Setting this property replaces all the children with the parsed contents of the given string.

InnerXml

ToString

[C#] public virtual string InnerXml {get; set;}

[C++] public: \_\_property virtual String\* get\_InnerXml();public: \_\_property

virtual void set\_InnerXml(String\*);

[VB] Overridable Public Property InnerXml As String

1 [JScript] public function get InnerXml() : String; public function set  
2 InnerXml(String);

3  
4 *Description*

5 Gets or sets the markup representing just the children of this node.

6 This is also settable which replaces the children of the node with the parsed  
7 contents of the given string. The parsing is done in the current namespace context.

8 IsReadOnly

9 ToString

10  
11 [C#] public virtual bool IsReadOnly {get;}

12 [C++] public: \_\_property virtual bool get\_IsReadOnly();

13 [VB] Overridable Public ReadOnly Property IsReadOnly As Boolean

14 [JScript] public function get IsReadOnly() : Boolean;

15  
16 *Description*

17 Gets a value indicating whether the node is read-only.

18 A read-only node is one whose properties, attributes, or children cannot be  
19 changed. You can remove a read-only node from the tree and insert it somewhere  
20 else. For example, Entity nodes are always read-only.

21 Item

22 ToString

23  
24 [C#] public virtual XmlElement this[string name] {get;}

25 [C++] public: \_\_property virtual XmlElement\* get\_Item(String\* name);

[VB] Overridable Public Default ReadOnly Property Item(ByVal name As String)  
As XmlElement  
[JScript] returnValue = XmlNodeObject.Item(name); Gets the specified child  
element.

### *Description*

Gets the first child element with the specified

**System.Xml.XmlNode.Name** . The qualified name of the element to retrieve

Item

ToString

[C#] public virtual XmlElement this[string localname, string ns] {get;}

[C++] public: \_\_property virtual XmlElement\* get\_Item(String\* localname,  
String\* ns);

[VB] Overridable Public Default ReadOnly Property Item(ByVal localname As  
String, ByVal ns As String) As XmlElement

[JScript] returnValue = XmlNodeObject.Item(localname, ns);

### *Description*

Gets the first child element with the specified

**System.Xml.XmlNode.LocalName** and **System.Xml.XmlNode.NamespaceURI**

. The local name of the element. The namespace URI of the element.

LastChild

ToString

[C#] public virtual XmlNode LastChild {get;}

[C++] public: \_\_property virtual XmlNode\* get\_LastChild();

[VB] Overridable Public ReadOnly Property LastChild As XmlNode

[JScript] public function get LastChild() : XmlNode;

### *Description*

Gets the last child of the node.

LocalName

ToString

[C#] public abstract string LocalName {get;}

[C++] public: \_\_property virtual String\* get\_LocalName() = 0;

[VB] MustOverride Public ReadOnly Property LocalName As String

[JScript] public abstract function get LocalName() : String;

### *Description*

When overridden in a derived class, gets the local name of the node.

If the node does not have a prefix, LocalName is the same as

### **System.Xml.XmlNode.Name .**

Name

ToString

[C#] public abstract string Name {get;}

[C++] public: \_\_property virtual String\* get\_Name() = 0;

1 [VB] MustOverride Public ReadOnly Property Name As String

2 [JScript] public abstract function get Name() : String;

3  
4 *Description*

5 When overridden in a derived class, gets the qualified name of the node.

6 NamespaceURI

7 ToString

8  
9 [C#] public virtual string NamespaceURI {get;}

10 [C++] public: \_\_property virtual String\* get\_NamespaceURI();

11 [VB] Overridable Public ReadOnly Property NamespaceURI As String

12 [JScript] public function get NamespaceURI() : String;

13  
14 *Description*

15 Gets the namespace URI of this node.

16 This is the namespace URI specified at creation time. For example,

17 **NamespaceURI** is urn:samples for the element An attribute does not inherit its  
18 namespace from the element it is attached to. If an attribute is not explicitly given  
19 a namespace, it simply has no namespace.

20 NextSibling

21 ToString

22  
23 [C#] public virtual XmlNode NextSibling {get;}

24 [C++] public: \_\_property virtual XmlNode\* get\_NextSibling();

25 [VB] Overridable Public ReadOnly Property NextSibling As XmlNode

1 [JScript] public function get NextSibling() : XmlNode;

3 *Description*

4 Gets the node immediately following this node.

5 NodeType

6 ToString

8 [C#] public abstract XmlNodeType NodeType {get;}

9 [C++] public: \_\_property virtual XmlNodeType get \_NodeType() = 0;

10 [VB] MustOverride Public ReadOnly Property NodeType As XmlNodeType

11 [JScript] public abstract function get NodeType() : XmlNodeType;

13 *Description*

14 When overridden in a derived class, gets the type of the current node.

15 This property never returns the **XmlNodeType** EndElement, EndEntity or

16 None.

17 OuterXml

18 ToString

20 [C#] public virtual string OuterXml {get;}

21 [C++] public: \_\_property virtual String\* get \_OuterXml();

22 [VB] Overridable Public ReadOnly Property OuterXml As String

23 [JScript] public function get OuterXml() : String;

25 *Description*



Gets the markup representing this node and all its children.

OwnerDocument

ToString

[C#] public virtual XmlDocument OwnerDocument {get;}

[C++] public: \_\_property virtual XmlDocument\* get\_OwnerDocument();

[VB] Overridable Public ReadOnly Property OwnerDocument As XmlDocument

[JScript] public function get OwnerDocument() : XmlDocument;

### *Description*

Gets the **System.Xml.XmlDocument** to which this node belongs.

When adding nodes to the current node, use the

**System.Xml.XmlDocument** returned by the

**System.Xml.XmlNode.OwnerDocument** property to create the node.

ParentNode

ToString

[C#] public virtual XmlNode ParentNode {get;}

[C++] public: \_\_property virtual XmlNode\* get\_ParentNode();

[VB] Overridable Public ReadOnly Property ParentNode As XmlNode

[JScript] public function get ParentNode() : XmlNode;

### *Description*

Gets the parent of this node (for nodes that can have parents).

All nodes except **System.Xml.XmlDocument** ,  
**System.Xml.XmlDocumentFragment** , and **System.Xml.XmlAttribute** can  
have a parent. However, if a node has just been created and not yet added to the  
tree, or if it has been removed from the tree, the parent is **null** .

Prefix

ToString

[C#] public virtual string Prefix {get; set;}

[C++] public: \_\_property virtual String\* get\_Prefix();public: \_\_property virtual  
void set\_Prefix(String\*);

[VB] Overridable Public Property Prefix As String

[JScript] public function get Prefix() : String;public function set Prefix(String);

### *Description*

Gets or sets the namespace prefix of this node.

For example, **Prefix** is bk for the element Setting this property, when  
permitted, changes the name property, which holds the qualified name, as well as  
the TagName on XmlElement and the and Name properties on XmlAttribute.  
Setting this property on node types that cannot have a prefix (such as Text,  
Comment, EntityReference, CDATA, ProcessingInstruction, Document, and  
DocumentFragment) has no effect. Note also that changing the prefix of an  
attribute that is known to have a default value, does not make a new attribute with  
the default value and the original prefix appear, since the namespaceURI and  
localName do not change.

PreviousSibling

## ToString

```
[C#] public virtual XmlNode PreviousSibling {get;}
[C++] public: __property virtual XmlNode* get_PreviousSibling();
[VB] Overridable Public ReadOnly Property PreviousSibling As XmlNode
[JScript] public function get PreviousSibling() : XmlNode;
```

### Description

Gets the node immediately preceding this node.

### Value

### ToString

```
[C#] public virtual string Value {get; set;}
[C++] public: __property virtual String* get_Value();public: __property virtual
void set_Value(String*);
[VB] Overridable Public Property Value As String
[JScript] public function get Value() : String;public function set Value(String);
```

### Description

Gets or sets the value of the node.

### AppendChild

```
[C#] public virtual XmlNode AppendChild(XmlNode newChild);
[C++] public: virtual XmlNode* AppendChild(XmlNode* newChild);
[VB] Overridable Public Function AppendChild(ByVal newChild As XmlNode)
```

1 As XmlNode

2 [JScript] public function AppendChild(newChild : XmlNode) : XmlNode;

3  
4 *Description*

5 Adds the specified node to the end of the list of children of this node.

6 *Return Value:* The node added.

7 If the *newChild* is already in the tree, it is first removed. The node to add. If  
8 it is a XmlDocumentFragment, the entire contents of the document fragment are  
9 moved into the child list of this node.

10 Clone

11  
12 [C#] public virtual XmlNode Clone();

13 [C++] public: virtual XmlNode\* Clone();

14 [VB] Overridable Public Function Clone() As XmlNode

15 [JScript] public function Clone() : XmlNode;

16  
17 *Description*

18 Creates a duplicate of this node.

19 *Return Value:* The cloned node.

20 Cloning an **System.Xml.XmlElement** copies all attributes and their values,  
21 including those generated by the XML processor to represent defaulted attributes.  
22 This method recursively clones the node and the subtree underneath it.

23 CloneNode

24  
25 [C#] public abstract XmlNode CloneNode(bool deep);

1 [C++] public: virtual XmlNode\* CloneNode(bool deep) = 0;

2 [VB] MustOverride Public Function CloneNode(ByVal deep As Boolean) As

3 XmlNode

4 [JScript] public abstract function CloneNode(deep : Boolean) : XmlNode;

6 *Description*

7 When overridden in a derived class, creates a duplicate of the node.

8 *Return Value:* The cloned node.

9 This method serves as a copy constructor for nodes. The duplicate node has  
10 no parent ( **System.Xml.XmlNode.ParentNode** returns **null** ). **true** to recursively  
11 clone the subtree under the specified node; **false** to clone only the node itself.

12 *CreateNavigator*

14 [C#] public XPathNavigator CreateNavigator();

15 [C++] public: \_\_sealed XPathNavigator\* CreateNavigator();

16 [VB] NotOverridable Public Function CreateNavigator() As XPathNavigator

17 [JScript] public function CreateNavigator() : XPathNavigator;

19 *Description*

20 Creates an **System.Xml.XPath.XPathNavigator** for navigating this object.

21 *Return Value:* An **XPathNavigator** object. The **XPathNavigator** is positioned on  
22 the node from which the method was called. It is not positioned on the root of the  
23 document.

24 The **XPathNavigator** provides read-only, random access to data. Because  
25 it is optimized for XSLT transformations, it provides performance benefits when

used as an input mechanism to the

**System.Xml.Xsl.XslTransform.Transform(System.Xml.XPath.IXPathNavigable, System.Xml.Xsl.XsltArgumentList)** method.

GetEnumerator

[C#] public IEnumerator GetEnumerator();

[C++] public: IEnumerator\* GetEnumerator();

[VB] Public Function GetEnumerator() As IEnumerator

[JScript] public function GetEnumerator() : IEnumerator;

### *Description*

Provides support for the for each style iteration over the nodes in the

**XmlNode** .

*Return Value:* An **System.Collections.IEnumerator** .

GetNamespaceOfPrefix

[C#] public virtual string GetNamespaceOfPrefix(string prefix);

[C++] public: virtual String\* GetNamespaceOfPrefix(String\* prefix);

[VB] Overridable Public Function GetNamespaceOfPrefix(ByVal prefix As String) As String

[JScript] public function GetNamespaceOfPrefix(prefix : String) : String;

### *Description*

Looks up the closest xmlns declaration for the given prefix that is in scope for the current node and returns the namespace URI in the declaration.

*Return Value:* The namespace URI of the specified prefix. Prefix whose namespace URI you want to find

### GetPrefixOfNamespace

[C#] public virtual string GetPrefixOfNamespace(string namespaceURI);

[C++] public: virtual String\* GetPrefixOfNamespace(String\* namespaceURI);

[VB] Overridable Public Function GetPrefixOfNamespace(ByVal namespaceURI As String) As String

[JScript] public function GetPrefixOfNamespace(namespaceURI : String) : String;

### Description

Looks up the closest xmlns declaration for the given namespace URI that is in scope for the current node and returns the prefix defined in that declaration.

*Return Value:* The prefix for the specified namespace URI. Namespace URI whose prefix you want to find

### InsertAfter

[C#] public virtual XmlNode InsertAfter(XmlNode newChild, XmlNode refChild);

[C++] public: virtual XmlNode\* InsertAfter(XmlNode\* newChild, XmlNode\* refChild);

[VB] Overridable Public Function InsertAfter(ByVal newChild As XmlNode, ByVal refChild As XmlNode) As XmlNode

[JScript] public function InsertAfter(newChild : XmlNode, refChild : XmlNode) : XmlNode;

## Description

Inserts the specified node immediately after the specified reference node.

If *refChild* is null, insert *newChild* at the beginning of the list of children. If *newChild* is a XmlDocumentFragment object, all of its children are inserted, in the same order, after *refChild* . If the *newChild* is already in the tree, it is first removed. XmlNode to insert. XmlNode that is the reference node. The newNode is placed after the refNode.

## InsertBefore

```
[C#] public virtual XmlNode InsertBefore(XmlNode newChild, XmlNode  
refChild);
```

```
[C++] public: virtual XmlNode* InsertBefore(XmlNode* newChild, XmlNode*  
refChild);
```

```
[VB] Overridable Public Function InsertBefore(ByVal newChild As XmlNode,  
ByVal refChild As XmlNode) As XmlNode
```

```
[JScript] public function InsertBefore(newChild : XmlNode, refChild : XmlNode)  
: XmlNode;
```

## Description

Inserts the specified node immediately before the specified reference node.

If *refChild* is **null** , insert *newChild* at the end of the list of children. If *newChild* is a XmlDocumentFragment object, all of its children are inserted, in the same order, before *refChild* . If the *newChild* is already in the tree, it is first



removed. The **XmlNode** to insert. The **XmlNode** that is the reference node. The *newChild* is placed before this node.

### Normalize

[C#] public virtual void Normalize();

[C++] public: virtual void Normalize();

[VB] Overridable Public Sub Normalize()

[JScript] public function Normalize();

### Description

Puts all **XmlText** nodes in the full depth of the sub-tree underneath this **XmlNode** into a "normal" form where only markup (e.g., tags, comments, processing instructions, CDATA sections, and entity references) separates **XmlText** nodes, that is, there are no adjacent **XmlText** nodes.

This method can be used to ensure that the DOM view of a document is the same as if it were saved and re-loaded, and is useful when operations (such as **XPointer** lookups) that depend on a particular document tree structure are to be used.

### PrependChild

[C#] public virtual XmlNode PrependChild(XmlNode newChild);

[C++] public: virtual XmlNode\* PrependChild(XmlNode\* newChild);

[VB] Overridable Public Function PrependChild(ByVal newChild As XmlNode)

As XmlNode

[JScript] public function PrependChild(newChild : XmlNode) : XmlNode;

## Description

Adds the specified node to the beginning of the list of children of this node.

*Return Value:* The node added.

If the *newChild* is already in the tree, it is first removed. The node to add. If it is a *XmlDocumentFragment*, the entire contents of the document fragment are moved into the child list of this node.

## RemoveAll

[C#] public virtual void RemoveAll();

[C++] public: virtual void RemoveAll();

[VB] Overridable Public Sub RemoveAll()

[JScript] public function RemoveAll();

## Description

Removes all the children and/or attributes of the current node.

If a removed attribute is known to have a default value, an attribute immediately appears containing the default value and, if applicable, the corresponding namespace URI, local name, and prefix.

## RemoveChild

[C#] public virtual XmlNode RemoveChild(XmlNode oldChild);

[C++] public: virtual XmlNode\* RemoveChild(XmlNode\* oldChild);

[VB] Overridable Public Function RemoveChild(ByVal oldChild As XmlNode)

As XmlNode

[JScript] public function RemoveChild(oldChild : XmlNode) : XmlNode;

### Description

Removes specified child node.

*Return Value:* The node removed.

When overriding **RemoveChild** in a derived class, in order for events to be fired correctly, you must call the base class's **RemoveChild** method. The node being removed.

### ReplaceChild

[C#] public virtual XmlNode ReplaceChild(XmlNode newChild, XmlNode oldChild);

[C++] public: virtual XmlNode\* ReplaceChild(XmlNode\* newChild, XmlNode\* oldChild);

[VB] Overridable Public Function ReplaceChild(ByVal newChild As XmlNode, ByVal oldChild As XmlNode) As XmlNode

[JScript] public function ReplaceChild(newChild : XmlNode, oldChild : XmlNode) : XmlNode;

### Description

Replaces the child node *oldChild* with *newChild* node.

*Return Value:* The node replaced.

If the *newChild* is already in the tree, it is first removed. The new node to put in the child list. The node being replaced in the list.

### SelectNodes

```

1
2 [C#] public XmlNodeList SelectNodes(string xpath);
3 [C++] public: XmlNodeList* SelectNodes(String* xpath);
4 [VB] Public Function SelectNodes(ByVal xpath As String) As XmlNodeList
5 [JScript] public function SelectNodes(xpath : String) : XmlNodeList; Selects a list
6 of nodes matching the XPath expression.

```

### *Description*

Selects a list of nodes matching the XPath expression.

*Return Value:* An **System.Xml.XmlNodeList** containing a collection of nodes matching the XPath query.

If the XPath expression requires namespace resolution, you must use the **SelectNodes** overload which takes an **System.Xml.XmlNamespaceManager** as its argument. The **XmlNamespaceManager** is used to resolve namespaces. The XPath expression.

### *SelectNodes*

```

17
18 [C#] public XmlNodeList SelectNodes(string xpath, XmlNamespaceManager
19 nsmgr);
20 [C++] public: XmlNodeList* SelectNodes(String* xpath,
21 XmlNamespaceManager* nsmgr);
22 [VB] Public Function SelectNodes(ByVal xpath As String, ByVal nsmgr As
23 XmlNamespaceManager) As XmlNodeList
24 [JScript] public function SelectNodes(xpath : String, nsmgr :
25 XmlNamespaceManager) : XmlNodeList;

```

## Description

Selects a list of nodes matching the XPath expression. Any prefixes found in the XPath expression are resolved using the supplied

**System.Xml.XmlNamespaceManager**.

*Return Value:* An **System.Xml.XmlNodeList** containing a collection of nodes matching the XPath query.

XPath expressions can include namespaces. Namespace resolution is supported using the **XmlNamespaceManager**. If the XPath expression includes a prefix, the prefix and namespace URI pair must be added to the **XmlNamespaceManager**. The XPath expression. An **System.Xml.XmlNamespaceManager** to use for resolving namespaces for prefixes in the XPath expression.

## SelectSingleNode

[C#] public XmlNode SelectSingleNode(string xpath);

[C++] public: XmlNode\* SelectSingleNode(String\* xpath);

[VB] Public Function SelectSingleNode(ByVal xpath As String) As XmlNode

[JScript] public function SelectSingleNode(xpath : String) : XmlNode; Selects the first **XmlNode** that matches the XPath expression.

## Description

Selects the first **XmlNode** that matches the XPath expression.

*Return Value:* The first **XmlNode** that matches the XPath query or **null** if no matching node was found.

If the XPath expression requires namespace resolution, you must use the **SelectSingleNode** overload which takes an **System.Xml.XmlNamespaceManager** as its argument. The **XmlNamespaceManager** is used to resolve namespaces. The XPath expression.

SelectSingleNode

```
[C#] public XmlNode SelectSingleNode(string xpath, XmlNamespaceManager
nsmgr);
[C++] public: XmlNode* SelectSingleNode(String* xpath,
XmlNamespaceManager* nsmgr);
[VB] Public Function SelectSingleNode(ByVal xpath As String, ByVal nsmgr As
XmlNamespaceManager) As XmlNode
[JScript] public function SelectSingleNode(xpath : String, nsmgr :
XmlNamespaceManager) : XmlNode;
```

### *Description*

Selects the first **XmlNode** that matches the XPath expression. Any prefixes found in the XPath expression are resolved using the supplied **System.Xml.XmlNamespaceManager**.

*Return Value:* The first **XmlNode** that matches the XPath query or **null** if no matching node was found.

XPath expressions can include namespaces. Namespace resolution is supported using the **XmlNamespaceManager**. If the XPath expression includes a prefix, the prefix and namespace URI pair must be added to the **XmlNamespaceManager**. The XPath expression. An

1 **System.Xml.XmlNamespaceManager** to use for resolving namespaces for  
2 prefixes in the XPath expression.

### 3 Supports

4  
5 [C#] public virtual bool Supports(string feature, string version);  
6 [C++] public: virtual bool Supports(String\* feature, String\* version);  
7 [VB] Overridable Public Function Supports(ByVal feature As String, ByVal  
8 version As String) As Boolean  
9 [JScript] public function Supports(feature : String, version : String) : Boolean;

### 11 *Description*

12 Test if the DOM implementation implements a specific feature.

13 *Return Value:* **true** if the feature is implemented in the specified version;  
14 otherwise, **false** . The package name of the feature to test. This name is case-  
15 insensitive. This is the version number of the package name to test. If the version  
16 is not specified (null), supporting any version of the feature will cause the method  
17 to return true.

### 18 IEnumerable.GetEnumerator

19  
20 [C#] IEnumerator IEnumerable.GetEnumerator();  
21 [C++] IEnumerator\* IEnumerable::GetEnumerator();  
22 [VB] Function GetEnumerator() As IEnumerator Implements  
23 IEnumerable.GetEnumerator  
24 [JScript] function IEnumerable.GetEnumerator() : IEnumerator;

### 25 ICloneable.Clone

1  
2 [C#] object ICloneable.Clone();

3 [C++] Object\* ICloneable::Clone();

4 [VB] Function Clone() As Object Implements ICloneable.Clone

5 [JScript] function ICloneable.Clone() : Object;

6       WriteContentTo

7  
8 [C#] public abstract void WriteContentTo(XmlWriter w);

9 [C++] public: virtual void WriteContentTo(XmlWriter\* w) = 0;

10 [VB] MustOverride Public Sub WriteContentTo(ByVal w As XmlWriter)

11 [JScript] public abstract function WriteContentTo(w : XmlWriter);

12  
13 *Description*

14       When overridden in a derived class, saves all the children of the node to the  
15 specified **System.Xml.XmlWriter** .

16       This method is functionally equivalent to the  
17 **System.Xml.XmlNode.InnerXml** property. The **XmlWriter** to which you want  
18 to save.

19       WriteTo

20  
21 [C#] public abstract void WriteTo(XmlWriter w);

22 [C++] public: virtual void WriteTo(XmlWriter\* w) = 0;

23 [VB] MustOverride Public Sub WriteTo(ByVal w As XmlWriter)

24 [JScript] public abstract function WriteTo(w : XmlWriter);



1  
2 *Description*

3 When overridden in a derived class, saves the current node to the specified

4 **System.Xml.XmlWriter** .

5 This method is functionally equivalent to the

6 **System.Xml.XmlNode.OuterXml** property. The **XmlWriter** to which you want  
7 to save.

8 XmlNodeChangedAction enumeration (System.Xml)

9 WriteTo

10  
11  
12 *Description*

13 TODO: Specifies the type of node change TODO: Specifies the type of  
14 node change

15 WriteTo

16  
17 [C#] public const XmlNodeChangedAction Change;

18 [C++] public: const XmlNodeChangedAction Change;

19 [VB] Public Const Change As XmlNodeChangedAction

20 [JScript] public var Change : XmlNodeChangedAction;

21  
22 *Description*

23 TODO: A node value is beeing changed.

24 WriteTo

1  
2 [C#] public const XmlNodeChangedAction Insert;

3 [C++] public: const XmlNodeChangedAction Insert;

4 [VB] Public Const Insert As XmlNodeChangedAction

5 [JScript] public var Insert : XmlNodeChangedAction;

6  
7 *Description*

8       TODO: A node is beeing inserted in the tree.

9       WriteTo

10  
11 [C#] public const XmlNodeChangedAction Remove;

12 [C++] public: const XmlNodeChangedAction Remove;

13 [VB] Public Const Remove As XmlNodeChangedAction

14 [JScript] public var Remove : XmlNodeChangedAction;

15  
16 *Description*

17       TODO: A node is beeing removed from the tree.

18       XmlNodeChangedEventArgs class (System.Xml)

19       ToString

20  
21  
22 *Description*

23       Provides data for the **System.Xml.XmlDocument.NodeChanged** ,

24 **System.Xml.XmlDocument.NodeChanging** ,

25 **System.Xml.XmlDocument.NodeInserted** ,

1 **System.Xml.XmlDocument.NodeInserting** ,  
2 **System.Xml.XmlDocument.NodeRemoved** and  
3 **System.Xml.XmlDocument.NodeRemoving** events.

4 The following C# code shows how to use the event handler.

5 Action

6 ToString

7  
8 [C#] public XmlNodeChangedAction Action {get;}

9 [C++] public: \_\_property XmlNodeChangedAction get \_Action();

10 [VB] Public ReadOnly Property Action As XmlNodeChangedAction

11 [JScript] public function get Action() : XmlNodeChangedAction;

12  
13 *Description*

14 Gets a value indicating what type of node change event is occurring.

15 NewParent

16 ToString

17  
18 [C#] public XmlNode NewParent {get;}

19 [C++] public: \_\_property XmlNode\* get \_NewParent();

20 [VB] Public ReadOnly Property NewParent As XmlNode

21 [JScript] public function get NewParent() : XmlNode;

22  
23 *Description*

24 Gets the value of the **System.Xml.XmlNode.ParentNode** after the  
25 operation completes.

```

1      Node
2      ToString
3
4  [C#] public XmlNode Node {get;}
5  [C++] public: __property XmlNode* get_Node();
6  [VB] Public ReadOnly Property Node As XmlNode
7  [JScript] public function get Node() : XmlNode;
8

```

#### *Description*

Gets the **System.Xml.XmlNode** that is being added, removed or changed.

OldParent

ToString

```

13
14 [C#] public XmlNode OldParent {get;}
15 [C++] public: __property XmlNode* get_OldParent();
16 [VB] Public ReadOnly Property OldParent As XmlNode
17 [JScript] public function get OldParent() : XmlNode;
18

```

#### *Description*

Gets the value of the **System.Xml.XmlNode.ParentNode** before the operation began.

XmlNodeChangedEventHandler delegate (System.Xml)

ToString

```

24
25

```

### *Description*

Represents the method that handles **System.Xml.XmlDocument.NodeChanged** , **System.Xml.XmlDocument.NodeChanging** , **System.Xml.XmlDocument.NodeInserted** , **System.Xml.XmlDocument.NodeInserting** , **System.Xml.XmlDocument.NodeRemoved** and **System.Xml.XmlDocument.NodeRemoving** events. The source of the event. An **System.Xml.XmlNodeChangedEventArgs** containing the event data.

When you create an **XmlNodeChangedEventHandler** delegate, you identify the method that handles the event. To associate the event with your event handler, add an instance of the delegate to the event. The event handler is called whenever the event occurs, unless you remove the delegate. For more information about event handler delegates, see .

**XmlNodeList** class (System.Xml)

**ToString**

### *Description*

Represents an ordered collection of nodes.

The **XmlNodeList** is "live": changes to the children of the node object that it was created from are immediately reflected in the nodes returned by the **XmlNodeList** properties and methods. The **XmlNodeList** is not a static snapshot

of the content of the node. This is true for every **XmlNodeList** , including the ones returned by the **System.Xml.XmlDocument.GetElementsByTagName(System.String)** method.

**XmlNodeList**

*Example Syntax:*

**ToString**

[C#] protected XmlNodeList();

[C++] protected: XmlNodeList();

[VB] Protected Sub New()

[JScript] protected function XmlNodeList();

**Count**

**ToString**

[C#] public abstract int Count {get;}

[C++] public: \_\_property virtual int get\_Count() = 0;

[VB] MustOverride Public ReadOnly Property Count As Integer

[JScript] public abstract function get Count() : int;

### *Description*

Gets the number of nodes in the XmlNodeList.

**ItemOf**

**ToString**

[C#] public virtual XmlNode this[int i] {get;}

```

1 [C++] public: __property virtual XmlNode* get_ItemOf(int i);
2 [VB] Overridable Public Default ReadOnly Property ItemOf(ByVal i As Integer)
3 As XmlNode
4 [JScript] returnValue = XmlNodeListObject.ItemOf(i);
5

```

### *Description*

Retrieves a node at the given index. Zero-based index into the list of nodes.

GetEnumerator

```

10 [C#] public abstract IEnumerator GetEnumerator();
11 [C++] public: virtual IEnumerator* GetEnumerator() = 0;
12 [VB] MustOverride Public Function GetEnumerator() As IEnumerator
13 [JScript] public abstract function GetEnumerator() : IEnumerator;
14

```

### *Description*

Provides a simple "foreach" style iteration over the collection of nodes in the **XmlNodeList** .

*Return Value:* An **System.Collections.IEnumerator** .

Item

```

21 [C#] public abstract XmlNode Item(int index);
22 [C++] public: virtual XmlNode* Item(int index) = 0;
23 [VB] MustOverride Public Function Item(ByVal index As Integer) As XmlNode
24 [JScript] public abstract function Item(index : int) : XmlNode;
25

```

## *Description*

Retrieves a node at the given index.

*Return Value:* The **System.Xml.XmlNode** in the collection. If *index* is greater than or equal to the number of nodes in the list, this returns **null** . Zero-based index into the list of nodes.

XmlNodeOrder enumeration (System.Xml)

ToString

## *Description*

Describes the document order of a node compared to a second node.

ToString

[C#] public const XmlNodeOrder After;

[C++] public: const XmlNodeOrder After;

[VB] Public Const After As XmlNodeOrder

[JScript] public var After : XmlNodeOrder;

## *Description*

The current node of the supplied navigator is after the current node of this navigator.

ToString

[C#] public const XmlNodeOrder Before;



[C++] public: const XmlNodeOrder Before;  
 [VB] Public Const Before As XmlNodeOrder  
 [JScript] public var Before : XmlNodeOrder;

*Description*

The current node of the supplied navigator is before the current node of this navigator.

ToString

[C#] public const XmlNodeOrder Same;  
 [C++] public: const XmlNodeOrder Same;  
 [VB] Public Const Same As XmlNodeOrder  
 [JScript] public var Same : XmlNodeOrder;

*Description*

The current node of the supplied navigator is the same as the current node of this navigator.

ToString

[C#] public const XmlNodeOrder Unknown;  
 [C++] public: const XmlNodeOrder Unknown;  
 [VB] Public Const Unknown As XmlNodeOrder  
 [JScript] public var Unknown : XmlNodeOrder;

*Description*

The node positions cannot be determined in document order, relative to each other. This could occur if the two nodes reside in different trees.

**XmlNodeReader** class (System.Xml)

**ToString**

### *Description*

Represents a reader that provides fast, non-cached forward only access to XML data in an **System.Xml.XmlNode**.

The **XmlNodeReader** has the ability to read an XML DOM subtree. This class does not support DTD or schema validation. To perform data validation, use the **System.Xml.XmlValidatingReader**.

**XmlNodeReader**

*Example Syntax:*

**ToString**

[C#] public XmlNodeReader(XmlNode node);

[C++] public: XmlNodeReader(XmlNode\* node);

[VB] Public Sub New(ByVal node As XmlNode)

[JScript] public function XmlNodeReader(node : XmlNode); Creates an instance of the **XmlNodeReader**.

### *Description*

Creates an instance of the **XmlNodeReader** class using the specified **System.Xml.XmlNode**.

The following C# code creates an **XmlNodeReader** . The **XmlNode** you want to read.

AttributeCount

ToString

[C#] public override int AttributeCount {get;}

[C++] public: \_\_property virtual int get\_AttributeCount();

[VB] Overrides Public ReadOnly Property AttributeCount As Integer

[JScript] public function get AttributeCount() : int;

#### *Description*

Gets the number of attributes on the current node.

This property is relevant to **Element** , **DocumentType** and **XmlDeclaration** nodes only. (Other node types do not have attributes.) The following example reads all the attributes on the root node.

BaseURI

ToString

[C#] public override string BaseURI {get;}

[C++] public: \_\_property virtual String\* get\_BaseURI();

[VB] Overrides Public ReadOnly Property BaseURI As String

[JScript] public function get BaseURI() : String;

#### *Description*

Gets the base URI of the current node.

A networked XML document is comprised of chunks of data aggregated using various W3C standard inclusion mechanisms and therefore contains nodes that come from different places. DTD entities are an example of this, but this is not limited to DTDs. The base URI tells you where these nodes came from. If there is no base URI for the nodes being returned (for example, they were parsed from an in-memory string), `String.Empty` is returned.

`CanResolveEntity`

`ToString`

[C#] public override bool `CanResolveEntity` {get;}

[C++] public: \_\_property virtual bool get\_`CanResolveEntity`();

[VB] Overrides Public ReadOnly Property `CanResolveEntity` As Boolean

[JScript] public function get `CanResolveEntity`() : Boolean;

### *Description*

Gets a value indicating whether this reader can parse and resolve entities.

`Depth`

`ToString`

[C#] public override int `Depth` {get;}

[C++] public: \_\_property virtual int get\_`Depth`();

[VB] Overrides Public ReadOnly Property `Depth` As Integer

[JScript] public function get `Depth`() : int;

### *Description*

Gets the depth of the current node in the XML document.

EOF

ToString

[C#] public override bool EOF {get;}

[C++] public: \_\_property virtual bool get\_EOF();

[VB] Overrides Public ReadOnly Property EOF As Boolean

[JScript] public function get EOF() : Boolean;

### *Description*

Gets a value indicating whether the reader is positioned at the end of the stream.

HasAttributes

ToString

[C#] public override bool HasAttributes {get;}

[C++] public: \_\_property virtual bool get\_HasAttributes();

[VB] Overrides Public ReadOnly Property HasAttributes As Boolean

[JScript] public function get HasAttributes() : Boolean;

### *Description*

Gets a value indicating whether the current node has any attributes.

HasValue

ToString

```

1
2 [C#] public override bool HasValue {get;}
3 [C++] public: __property virtual bool get_HasValue();
4 [VB] Overrides Public ReadOnly Property HasValue As Boolean
5 [JScript] public function get HasValue() : Boolean;
6

```

#### *Description*

Gets a value indicating whether the current node can have a

#### **System.Xml.XmlNodeReader.Value .**

The following table lists node types that have a value to return.

IsDefault

ToString

```

13
14 [C#] public override bool IsDefault {get;}
15 [C++] public: __property virtual bool get_IsDefault();
16 [VB] Overrides Public ReadOnly Property IsDefault As Boolean
17 [JScript] public function get IsDefault() : Boolean;
18

```

#### *Description*

Gets a value indicating whether the current node is an attribute that was generated from the default value defined in the DTD or schema.

This property applies to attribute nodes only.

IsEmptyElement

ToString

[C#] public override bool IsEmptyElement {get;}

[C++] public: \_\_property virtual bool get\_IsEmptyElement();

[VB] Overrides Public ReadOnly Property IsEmptyElement As Boolean

[JScript] public function get IsEmptyElement() : Boolean;

### Description

Gets a value indicating whether the current node is an empty element (for example, ).

This property enables you to determine the difference between the following: ( **IsEmptyElement** is **true** ).

Item

ToString

[C#] public override string this[int i] {get;}

[C++] public: \_\_property virtual String\* get\_Item(int i);

[VB] Overrides Public Default ReadOnly Property Item(ByVal i As Integer) As String

[JScript] returnValue = XmlNodeReaderObject.Item(i); Gets the value of the specified attribute.

### Description

Gets the value of the attribute with the specified index.

This property does not move the reader. The index of the attribute.

Item

ToString

[C#] public override string this[string name] {get;}

[C++] public: \_\_property virtual String\* get\_Item(String\* name);

[VB] Overrides Public Default ReadOnly Property Item(ByVal name As String)

As String

[JScript] returnValue = XmlNodeReaderObject.Item(name);

### Description

Gets the value of the attribute with the specified name.

This property does not move the reader. The qualified name of the attribute.

Item

ToString

[C#] public override string this[string name, string namespaceURI] {get;}

[C++] public: \_\_property virtual String\* get\_Item(String\* name, String\* namespaceURI);

[VB] Overrides Public Default ReadOnly Property Item(ByVal name As String,

ByVal namespaceURI As String) As String

[JScript] returnValue = XmlNodeReaderObject.Item(name, namespaceURI);

### Description

Gets the value of the attribute with the specified local name and namespace URI.



This property does not move the reader. The local name of the attribute.

The namespace URI of the attribute.

LocalName

ToString

[C#] public override string LocalName {get;}

[C++] public: \_\_property virtual String\* get\_LocalName();

[VB] Overrides Public ReadOnly Property LocalName As String

[JScript] public function get LocalName() : String;

### Description

Gets the local name of the current node.

Name

ToString

[C#] public override string Name {get;}

[C++] public: \_\_property virtual String\* get\_Name();

[VB] Overrides Public ReadOnly Property Name As String

[JScript] public function get Name() : String;

### Description

Gets the qualified name of the current node.

NamespaceURI

ToString

1  
2 [C#] public override string NamespaceURI {get;}

3 [C++] public: \_\_property virtual String\* get\_NamespaceURI();

4 [VB] Overrides Public ReadOnly Property NamespaceURI As String

5 [JScript] public function get NamespaceURI() : String;

6  
7 *Description*

8 Gets the namespace URI (as defined in the W3C Namespace specification)  
9 of the node on which the reader is positioned.

10 This property is relevant to **Element** and **Attribute** nodes only.

11 NameTable

12 ToString

13  
14 [C#] public override XmlNameTable NameTable {get;}

15 [C++] public: \_\_property virtual XmlNameTable\* get\_NameTable();

16 [VB] Overrides Public ReadOnly Property NameTable As XmlNameTable

17 [JScript] public function get NameTable() : XmlNameTable;

18  
19 *Description*

20 Gets the **System.Xml.XmlNameTable** associated with this  
21 implementation.

22 All node and attribute names returned from this class are atomized using  
23 the **NameTable** . When the same name is returned multiple times (for example,  
24 **Customer** ), then the same **String** object will be returned for that name. This  
25

1 makes it possible for you to write efficient code that does object comparisons on  
2 these strings instead of expensive string comparisons.

3       NodeType

4       ToString

5  
6 [C#] public override XmlNodeType NodeType {get;}

7 [C++] public: \_\_property virtual XmlNodeType get\_NodeType();

8 [VB] Overrides Public ReadOnly Property NodeType As XmlNodeType

9 [JScript] public function get NodeType() : XmlNodeType;

10  
11 *Description*

12       Gets the type of the current node.

13       Prefix

14       ToString

15  
16 [C#] public override string Prefix {get;}

17 [C++] public: \_\_property virtual String\* get\_Prefix();

18 [VB] Overrides Public ReadOnly Property Prefix As String

19 [JScript] public function get Prefix() : String;

20  
21 *Description*

22       Gets the namespace prefix associated with the current node.

23       QuoteChar

24       ToString

```

1
2 [C#] public override char QuoteChar {get;}
3 [C++] public: __property virtual __wchar_t get_QuoteChar();
4 [VB] Overrides Public ReadOnly Property QuoteChar As Char
5 [JScript] public function get QuoteChar() : Char;

```

### Description

Gets the quotation mark character used to enclose the value of an attribute node.

This property applies only to an attribute node.

ReadState

ToString

```

13
14 [C#] public override ReadState ReadState {get;}
15 [C++] public: __property virtual ReadState get_ReadState();
16 [VB] Overrides Public ReadOnly Property ReadState As ReadState
17 [JScript] public function get ReadState() : ReadState;

```

### Description

Gets the state of the reader.

Value

ToString

```

23
24 [C#] public override string Value {get;}
25 [C++] public: __property virtual String* get_Value();

```

[VB] Overrides Public ReadOnly Property Value As String

[JScript] public function get Value() : String;

### Description

Gets the text value of the current node.

XmlLang

ToString

[C#] public override string XmlLang {get;}

[C++] public: \_\_property virtual String\* get\_XmlLang();

[VB] Overrides Public ReadOnly Property XmlLang As String

[JScript] public function get XmlLang() : String;

### Description

Gets the current **xml:lang** scope.

This property represents the xml:lang scope within which the current node resides. For example, here is an XML fragment with **xml:lang** set to US English in the root element: Fred When the reader is positioned on the **name** element, you can use this property to find that it is in the scope of a US English **xml:lang** attribute.

XmlSpace

ToString

[C#] public override XmlSpace XmlSpace {get;}

[C++] public: \_\_property virtual XmlSpace get\_XmlSpace();

[VB] Overrides Public ReadOnly Property XmlSpace As XmlSpace

[JScript] public function get XmlSpace() : XmlSpace;

#### *Description*

Gets the current **xml:space** scope.

Close

[C#] public override void Close();

[C++] public: void Close();

[VB] Overrides Public Sub Close()

[JScript] public override function Close();

#### *Description*

Changes the **System.Xml.XmlNodeReader.ReadState** to Closed.

This method also releases any resources held while reading. If **Close** has already been called, no action is performed.

GetAttribute

[C#] public override string GetAttribute(int i);

[C++] public: String\* GetAttribute(int i);

[VB] Overrides Public Function GetAttribute(ByVal i As Integer) As String

[JScript] public override function GetAttribute(i : int) : String;

#### *Description*

Gets the value of the attribute with the specified index.

*Return Value:* The value of the specified attribute.

This method does not move the reader. The index of the attribute. The index is zero-based. (The first attribute has index 0.)

GetAttribute

[C#] public override string GetAttribute(string name);

[C++] public: String\* GetAttribute(String\* name);

[VB] Overrides Public Function GetAttribute(ByVal name As String) As String

[JScript] public override function GetAttribute(name : String) : String; Gets the value of an attribute.

### Description

Gets the value of the attribute with the specified name.

*Return Value:* The value of the specified attribute. If the attribute is not found, String.Empty is returned.

This method does not move the reader. The qualified name of the attribute.

GetAttribute

[C#] public override string GetAttribute(string name, string namespaceURI);

[C++] public: String\* GetAttribute(String\* name, String\* namespaceURI);

[VB] Overrides Public Function GetAttribute(ByVal name As String, ByVal namespaceURI As String) As String

[JScript] public override function GetAttribute(name : String, namespaceURI : String) : String;

## Description

Gets the value of the attribute with the specified local name and namespace URI.

*Return Value:* The value of the specified attribute. If the attribute is not found, `String.Empty` is returned.

The following XML contains an attribute in a specific namespace: You can lookup the **dt:type** attribute using one argument (prefix and local name) or two arguments (local name and namespace URI): `String dt = reader.GetAttribute("dt:type");` `String dt2 = reader.GetAttribute("type","urn:datatypes");` To lookup the **xmlns:dt** attribute, use one of the following arguments: `String dt3 = reader.GetAttribute("xmlns:dt");` `String dt4 = reader.GetAttribute("dt",http://www.w3.org/2000/xmlns/);` You can also get this information using the **System.Xml.XmlNodeReader.Prefix** property. The local name of the attribute. The namespace URI of the attribute.

## LookupNamespace

[C#] `public override string LookupNamespace(string prefix);`

[C++] `public: String* LookupNamespace(String* prefix);`

[VB] `Overrides Public Function LookupNamespace(ByVal prefix As String) As String`

[JScript] `public override function LookupNamespace(prefix : String) : String;`

## Description



Resolves a namespace prefix in the current element's scope.

*Return Value:* The namespace URI to which the prefix maps or **null** if no matching prefix is found.

In the preceding XML, if the reader is positioned on the **href** attribute, the prefix **a** is resolved by calling `reader.LookupNamesapce("a")` . The returned string is `urn:456` . The prefix whose namespace URI you want to resolve. To match the default namespace, pass an empty string. This string does not have to be atomized.

### MoveToAttribute

[C#] public override void MoveToAttribute(int i);

[C++] public: void MoveToAttribute(int i);

[VB] Overrides Public Sub MoveToAttribute(ByVal i As Integer)

[JScript] public override function MoveToAttribute(i : int);

### Description

Moves to the attribute with the specified index. The index of the attribute.

### MoveToAttribute

[C#] public override bool MoveToAttribute(string name);

[C++] public: bool MoveToAttribute(String\* name);

[VB] Overrides Public Function MoveToAttribute(ByVal name As String) As

Boolean

[JScript] public override function MoveToAttribute(name : String) : Boolean;

Moves to the specified attribute.

## Description

Moves to the attribute with the specified name.

*Return Value:* **true** if the attribute is found; otherwise, **false** . If **false** , the reader's position does not change.

After calling this method, the **System.Xml.XmlNodeReader.Name** , **System.Xml.XmlNodeReader.NamespaceURI** , and **System.Xml.XmlNodeReader.Prefix** properties reflect the properties of that attribute. The qualified name of the attribute.

## MoveToAttribute

[C#] public override bool MoveToAttribute(string name, string namespaceURI);

[C++] public: bool MoveToAttribute(String\* name, String\* namespaceURI);

[VB] Overrides Public Function MoveToAttribute(ByVal name As String, ByVal namespaceURI As String) As Boolean

[JScript] public override function MoveToAttribute(name : String, namespaceURI : String) : Boolean;

## Description

Moves to the attribute with the specified local name and namespace URI.

*Return Value:* **true** if the attribute is found; otherwise, **false** . If **false** , the reader's position does not change.

After calling this method, the **System.Xml.XmlNodeReader.Name** , **System.Xml.XmlNodeReader.NamespaceURI** , and

**System.Xml.XmlNodeReader.Prefix** properties reflect the properties of that attribute. The local name of the attribute. The namespace URI of the attribute.

### MoveToElement

[C#] public override bool MoveToElement();

[C++] public: bool MoveToElement();

[VB] Overrides Public Function MoveToElement() As Boolean

[JScript] public override function MoveToElement() : Boolean;

### Description

Moves to the element that contains the current attribute node.

**Return Value:** **true** if the reader is positioned on an attribute (the reader moves to the element that owns the attribute); **false** if the reader is not positioned on an attribute (the position of the reader does not change).

Use this method to return to an element after navigating through its attributes. This method moves the reader to one of the following node types:

**Element** , **DocumentType** , or **XmlDeclaration** .

### MoveToFirstAttribute

[C#] public override bool MoveToFirstAttribute();

[C++] public: bool MoveToFirstAttribute();

[VB] Overrides Public Function MoveToFirstAttribute() As Boolean

[JScript] public override function MoveToFirstAttribute() : Boolean;

### Description

Moves to the first attribute.

*Return Value:* **true** if an attribute exists (the reader moves to the first attribute); otherwise, **false** (the position of the reader does not change).

MoveToNextAttribute

[C#] public override bool MoveToNextAttribute();

[C++] public: bool MoveToNextAttribute();

[VB] Overrides Public Function MoveToNextAttribute() As Boolean

[JScript] public override function MoveToNextAttribute() : Boolean;

#### *Description*

Moves to the next attribute.

*Return Value:* **true** if there is a next attribute; **false** if there are no more attributes.

If the current node is an element node, this method is equivalent to

**System.Xml.XmlNodeReader.MoveToFirstAttribute** . If

**MoveToNextAttribute** returns **true** , the reader moves to the next attribute;

otherwise, the position of the reader does not change.

Read

[C#] public override bool Read();

[C++] public: bool Read();

[VB] Overrides Public Function Read() As Boolean

[JScript] public override function Read() : Boolean;

#### *Description*

Reads the next node from the stream.

*Return Value:* **true** if the next node was read successfully; **false** if there are no more nodes to read.

When a reader is first created and initialized, there is no information available. You must call **Read** to read the first node.

## ReadAttributeValue

[C#] public override bool ReadAttributeValue();

[C++] public: bool ReadAttributeValue();

[VB] Overrides Public Function ReadAttributeValue() As Boolean

[JScript] public override function ReadAttributeValue() : Boolean;

## Description

Parses the attribute value into one or more **Text** or **EntityReference** nodes.

*Return Value:* **true** if there are nodes to return.

Use this method after calling

**System.Xml.XmlNodeReader.MoveToAttribute(System.String)** to read through the text or entity reference nodes that make up the attribute value. The **System.Xml.XmlNodeReader.Depth** of the attribute value nodes is one plus the depth of the attribute node; it increments and decrements by one when you step into and out of general entity references.

## ReadInnerXml

[C#] public override string ReadInnerXml();

[C++] public: String\* ReadInnerXml();

1 [VB] Overrides Public Function ReadInnerXml() As String

2 [JScript] public override function ReadInnerXml() : String;

3  
4 *Description*

5 Reads all the content, including markup, as a string.

6 *Return Value:* All the XML content, including markup, in the current node. If the  
7 current node has no children, an empty string is returned.

8 This method returns all the content of the current node including the  
9 markup. The current node (start tag) and corresponding end node (end tag) are not  
10 returned. For example, if you had the following: this **ReadInnerXml** returns this  
11 This method handles element and attribute nodes in the following way: Node Type  
12 Child Content Return Value Position After the Call **Element** text text After the  
13 end tag.

14 **ReadOuterXml**

15  
16 [C#] public override string ReadOuterXml();

17 [C++] public: String\* ReadOuterXml();

18 [VB] Overrides Public Function ReadOuterXml() As String

19 [JScript] public override function ReadOuterXml() : String;

20  
21 *Description*

22 Reads the content, including markup, representing this node and all its  
23 children.

24 *Return Value:* If the reader is positioned on an element or an attribute node, this  
25

method returns all the XML content, including markup, of the current node and all its children; otherwise, it returns an empty string.

This method is similar to **System.Xml.XmlNodeReader.ReadInnerXml** except it also returns the start and end tags.

### ReadString

[C#] public override string ReadString();

[C++] public: String\* ReadString();

[VB] Overrides Public Function ReadString() As String

[JScript] public override function ReadString() : String;

### *Description*

Reads the contents of an element or text node as a string.

*Return Value:* The contents of the element or text node. This can be an empty string if the reader is positioned on something other than an element or text node, or if there is no more text content to return in the current context.

If positioned on an element, **ReadString** concatenates all text, significant whitespace, whitespace and CDATA section node types together and returns the concatenated data as the element content. It stops when any markup is encountered. This could occur in a mixed content model, or when an element end tag is read.

### ResolveEntity

[C#] public override void ResolveEntity();

[C++] public: void ResolveEntity();

1 [VB] Overrides Public Sub ResolveEntity()

2 [JScript] public override function ResolveEntity();

3  
4 *Description*

5 Resolves the entity reference for **EntityReference** nodes.

6 If the reader is positioned on an **EntityReference** node (  
7 **XmlNodeType.EntityReference**), if **System.Xml.XmlNodeReader.Read** is  
8 called after calling this method, the entity replacement text is parsed. When the  
9 entity replacement text is finished, an **EndEntity** node is returned to close the  
10 entity reference scope.

11 *Skip*

12  
13 [C#] public override void Skip();

14 [C++] public: void Skip();

15 [VB] Overrides Public Sub Skip()

16 [JScript] public override function Skip();

17  
18 *Description*

19 Skips the current element.

20 For example, suppose you have the following XML input: abc ... If the  
21 reader is positioned on the "" node or any of its attributes, calling **Skip** positions  
22 the reader to the "" **node**.

23 **XmlNodeType** enumeration (System.Xml)

24 **ToString**



**Description**

Specifies the type of node.

ToString

[C#] public const XmlNodeType Attribute;

[C++] public: const XmlNodeType Attribute;

[VB] Public Const Attribute As XmlNodeType

[JScript] public var Attribute : XmlNodeType;

**Description**

An attribute.

ToString

[C#] public const XmlNodeType CDATA;

[C++] public: const XmlNodeType CDATA;

[VB] Public Const CDATA As XmlNodeType

[JScript] public var CDATA : XmlNodeType;

**Description**

A CDATA section.

ToString

[C#] public const XmlNodeType Comment;

**[C++] public: const XmlNodeType Comment;**

**[VB] Public Const Comment As XmlNodeType**

**[JScript] public var Comment : XmlNodeType;**

***Description***

**A comment.**

ToString

**[C#] public const XmlNodeType Document;**

**[C++] public: const XmlNodeType Document;**

**[VB] Public Const Document As XmlNodeType**

**[JScript] public var Document : XmlNodeType;**

***Description***

**A document object that, as the root of the document tree, provides access to the entire XML document.**

ToString

**[C#] public const XmlNodeType DocumentFragment;**

**[C++] public: const XmlNodeType DocumentFragment;**

**[VB] Public Const DocumentFragment As XmlNodeType**

**[JScript] public var DocumentFragment : XmlNodeType;**

***Description***

**A document fragment.**

ToString

**[C#] public const XmlNodeType DocumentType;**

**[C++] public: const XmlNodeType DocumentType;**

**[VB] Public Const DocumentType As XmlNodeType**

**[JScript] public var DocumentType : XmlNodeType;**

***Description***

The document type declaration, indicated by the following tag.

ToString

**[C#] public const XmlNodeType Element;**

**[C++] public: const XmlNodeType Element;**

**[VB] Public Const Element As XmlNodeType**

**[JScript] public var Element : XmlNodeType;**

***Description***

An element.

ToString

**[C#] public const XmlNodeType EndElement;**

**[C++] public: const XmlNodeType EndElement;**

**[VB] Public Const EndElement As XmlNodeType**

**[JScript] public var EndElement : XmlNodeType;**

**Description**

An end element tag.

ToString

[C#] public const XmlNodeType EndEntity;

[C++] public: const XmlNodeType EndEntity;

[VB] Public Const EndEntity As XmlNodeType

[JScript] public var EndEntity : XmlNodeType;

**Description**

Returned when XmlReader gets to the end of the entity replacement as a result of a call to System.Xml.XmlReader.ResolveEntity .

ToString

[C#] public const XmlNodeType Entity;

[C++] public: const XmlNodeType Entity;

[VB] Public Const Entity As XmlNodeType

[JScript] public var Entity : XmlNodeType;

**Description**

An entity declaration.

ToString

[C#] public const XmlNodeType EntityReference;

1 **[C++] public: const XmlNodeType EntityReference;**  
2 **[VB] Public Const EntityReference As XmlNodeType**  
3 **[JScript] public var EntityReference : XmlNodeType;**

4  
5 ***Description***

6 **A reference to an entity.**

7 ToString

8  
9 **[C#] public const XmlNodeType None;**  
10 **[C++] public: const XmlNodeType None;**  
11 **[VB] Public Const None As XmlNodeType**  
12 **[JScript] public var None : XmlNodeType;**

13  
14 ***Description***

15 **This is returned by the System.Xml.XmlReader if a Read method has**  
16 **not been called.**

17 ToString

18  
19 **[C#] public const XmlNodeType Notation;**  
20 **[C++] public: const XmlNodeType Notation;**  
21 **[VB] Public Const Notation As XmlNodeType**  
22 **[JScript] public var Notation : XmlNodeType;**

23  
24 ***Description***

25 **A notation in the document type declaration.**

ToString

**[C#] public const XmlNodeType ProcessingInstruction;**  
**[C++] public: const XmlNodeType ProcessingInstruction;**  
**[VB] Public Const ProcessingInstruction As XmlNodeType**  
**[JScript] public var ProcessingInstruction : XmlNodeType;**

***Description***

**A processing instruction.**

ToString

**[C#] public const XmlNodeType SignificantWhitespace;**  
**[C++] public: const XmlNodeType SignificantWhitespace;**  
**[VB] Public Const SignificantWhitespace As XmlNodeType**  
**[JScript] public var SignificantWhitespace : XmlNodeType;**

***Description***

**Whitespace between markup in a mixed content model or whitespace within the xml:space="preserve" scope.**

ToString

**[C#] public const XmlNodeType Text;**  
**[C++] public: const XmlNodeType Text;**  
**[VB] Public Const Text As XmlNodeType**  
**[JScript] public var Text : XmlNodeType;**

**Description**

**The text content of a node.**

ToString

**[C#] public const XmlNodeType Whitespace;**

**[C++] public: const XmlNodeType Whitespace;**

**[VB] Public Const Whitespace As XmlNodeType**

**[JScript] public var Whitespace : XmlNodeType;**

**Description**

**Whitespace between markup.**

ToString

**[C#] public const XmlNodeType XmlDeclaration;**

**[C++] public: const XmlNodeType XmlDeclaration;**

**[VB] Public Const XmlDeclaration As XmlNodeType**

**[JScript] public var XmlDeclaration : XmlNodeType;**

**Description**

**The XML declaration.**

XmlNotation class (System.Xml)

ToString

***Description***

**Represents a notation declaration: .**

Attributes

BaseURI

ChildNodes

FirstChild

HasChildNodes

InnerText

InnerXml

ToString

***Description***

**Gets the markup representing the children of this node.**

**XmlNode nodes are read-only. Setting this property throws an exception.**

IsReadOnly

ToString

**[C#] public override bool IsReadOnly {get;}**

**[C++] public: \_\_property virtual bool get\_IsReadOnly();**

**[VB] Overrides Public ReadOnly Property IsReadOnly As Boolean**

**[JScript] public function get IsReadOnly() : Boolean;**



## **Description**

**Gets a value indicating whether the node is read-only.**

**A read-only node is one whose properties, attributes, or children cannot be changed. You can remove a read-only node from the tree and insert it somewhere else.**

Item

Item

LastChild

LocalName

ToString

## **Description**

**Gets the name of the current node without the namespace prefix.**

Name

ToString

**[C#] public override string Name {get;}**

**[C++] public: \_\_property virtual String\* get\_Name();**

**[VB] Overrides Public ReadOnly Property Name As String**

**[JScript] public function get Name() : String;**

## **Description**

**Gets the name of the current node.**

NamespaceURI

NextSibling

NodeType

ToString

### ***Description***

**Gets the type of the current node.**

OuterXml

ToString

**[C#] public override string OuterXml {get;}**

**[C++] public: \_\_property virtual String\* get\_OuterXml();**

**[VB] Overrides Public ReadOnly Property OuterXml As String**

**[JScript] public function get OuterXml() : String;**

### ***Description***

**Gets the markup representing this node and all its children.**

OwnerDocument

ParentNode

Prefix

PreviousSibling

PublicId

ToString

**Description**

**Gets the value of the public identifier on the notation declaration.**

SystemId

ToString

**[C#] public string SystemId {get;}**

**[C++] public: \_\_property String\* get\_SystemId();**

**[VB] Public ReadOnly Property SystemId As String**

**[JScript] public function get SystemId() : String;**

**Description**

**Gets the value of the system identifier on the notation declaration.**

Value

CloneNode

**[C#] public override XmlNode CloneNode(bool deep);**

**[C++] public: XmlNode\* CloneNode(bool deep);**

**[VB] Overrides Public Function CloneNode(ByVal deep As Boolean) As**

**XmlNode**

**[JScript] public override function CloneNode(deep : Boolean) : XmlNode;**

**Description**

Creates a duplicate of this node. Notation nodes cannot be cloned.

Calling this method on an XmlNode object throws an exception. true to recursively clone the subtree under the specified node; false to clone only the node itself.

WriteContentTo

[C#] public override void WriteContentTo(XmlWriter w);

[C++] public: void WriteContentTo(XmlWriter\* w);

[VB] Overrides Public Sub WriteContentTo(ByVal w As XmlWriter)

[JScript] public override function WriteContentTo(w : XmlWriter);

### *Description*

Saves the children of the node to the specified System.Xml.XmlWriter .

The XmlWriter to which you want to save.

WriteTo

[C#] public override void WriteTo(XmlWriter w);

[C++] public: void WriteTo(XmlWriter\* w);

[VB] Overrides Public Sub WriteTo(ByVal w As XmlWriter)

[JScript] public override function WriteTo(w : XmlWriter);

### *Description*

Saves the node to the specified System.Xml.XmlWriter . The XmlWriter to which you want to save.

XmlParserContext class (System.Xml)

WriteTo

**Description**

Provides all the context information required by System.Xml.XmlTextReader or System.Xml.XmlValidatingReader to parse an XML fragment.

XmlParserContext

**Example Syntax:**

WriteTo

**[C#] public XmlParserContext(XmlNameTable nt, XmlNamespaceManager nsMgr, string xmlLang, XmlSpace xmlSpace);**

**[C++] public: XmlParserContext(XmlNameTable\* nt, XmlNamespaceManager\* nsMgr, String\* xmlLang, XmlSpace xmlSpace);**

**[VB] Public Sub New(ByVal nt As XmlNameTable, ByVal nsMgr As XmlNamespaceManager, ByVal xmlLang As String, ByVal xmlSpace As XmlSpace)**

**[JScript] public function XmlParserContext(nt : XmlNameTable, nsMgr : XmlNamespaceManager, xmlLang : String, xmlSpace : XmlSpace);**

**Initializes a new instance of the XmlParserContext class with the specified values.**

**Description**

1        Initializes a new instance of the `XmlParserContext` class with the  
2 specified values.

3        If *nt* and the name table used to construct the *nsMgr* are not the same,  
4 `XmlParserContext` does not recognize namespaces added to *nsMgr* and  
5 throws an `System.Xml.XmlException` when it encounters the unrecognized  
6 namespaces. The `System.Xml.XmlNameTable` to use to atomize strings. If this  
7 is null, the name table used to construct the *nsMgr* is used instead. The  
8 `System.Xml.XmlNamespaceManager` to use for looking up namespace  
9 information. The `xml:lang` scope. An `System.Xml.XmlSpace` value indicating  
10 the `xml:space` scope.

11        `XmlParserContext`

12        *Example Syntax:*

13        `WriteTo`

14  
15        [C#] `public XmlParserContext(XmlNameTable nt, XmlNamespaceManager`  
16 `nsMgr, string docTypeName, string pubId, string sysId, string`  
17 `internalSubset, string baseURI, string xmlLang, XmlSpace xmlSpace);`

18        [C++] `public: XmlParserContext(XmlNameTable* nt,`  
19 `XmlNamespaceManager* nsMgr, String* docTypeName, String* pubId,`  
20 `String* sysId, String* internalSubset, String* baseURI, String* xmlLang,`  
21 `XmlSpace xmlSpace);`

22        [VB] `Public Sub New(ByVal nt As XmlNameTable, ByVal nsMgr As`  
23 `XmlNamespaceManager, ByVal docTypeName As String, ByVal pubId As`  
24 `String, ByVal sysId As String, ByVal internalSubset As String, ByVal`  
25 `baseURI As String, ByVal xmlLang As String, ByVal xmlSpace As`

1 XmlSpace)

2 [JScript] public function XmlParserContext(nt : XmlNameTable, nsMgr :  
3 XmlNamespaceManager, docTypeName : String, pubId : String, sysId :  
4 String, internalSubset : String, baseURI : String, xmlLang : String, xmlSpace  
5 : XmlSpace);

6  
7 **Description**

8       **Initializes a new instance of the XmlParserContext class with the**  
9 **specified values.**

10       **This constructor supplies all DocumentType information required by**  
11 **System.Xml.XmlValidatingReader . If this XmlParserContext is passed to an**  
12 **System.Xml.XmlTextReader , all DTD information is ignored. The**  
13 **System.Xml.XmlNameTable to use to atomize strings. If this is null, the name**  
14 **table used to construct the nsMgr is used instead. The**  
15 **System.Xml.XmlNamespaceManager to use for looking up namespace**  
16 **information. The name of the document type declaration. The public**  
17 **identifier. The system identifier. The internal DTD subset. The base URI for**  
18 **the XML fragment (the location from which the fragment was loaded). The**  
19 **xml:lang scope. An System.Xml.XmlSpace value indicating the xml:space**  
20 **scope.**

21       BaseURI

22       WriteTo

23  
24 [C#] public string BaseURI {get; set;}

25 [C++] public: \_\_property String\* get\_BaseURI();public: \_\_property void

1 **set\_BaseURI(String\*);**

2 **[VB] Public Property BaseURI As String**

3 **[JScript] public function get BaseURI() : String;public function set**

4 **BaseURI(String);**

6 ***Description***

7 **Gets or sets the base URI.**

8 **A networked XML document is comprised of chunks of data**  
 9 **aggregated using various W3C standard inclusion mechanisms and therefore**  
 10 **may contain nodes that come from different places.**

11 DocTypeName

12 WriteTo

14 **[C#] public string DocTypeName {get; set;}**

15 **[C++] public: \_\_property String\* get\_DocTypeName();public: \_\_property**

16 **void set\_DocTypeName(String\*);**

17 **[VB] Public Property DocTypeName As String**

18 **[JScript] public function get DocTypeName() : String;public function set**

19 **DocTypeName(String);**

21 ***Description***

22 **Gets or sets the name of the document type declaration.**

23 **This property, along with System.Xml.XmlParserContext.PublicId ,**  
 24 **System.Xml.XmlParserContext.SystemId , and**  
 25 **System.Xml.XmlParserContext.InternalSubset , provides all the document**



type declaration information. It is used to find and load the necessary DTD information so that you get all the default attributes and entities defined in the DTD.

InternalSubset

WriteTo

[C#] public string InternalSubset {get; set;}

[C++] public: \_\_property String\* get\_InternalSubset();public: \_\_property void set\_InternalSubset(String\*);

[VB] Public Property InternalSubset As String

[JScript] public function get InternalSubset() : String;public function set InternalSubset(String);

### *Description*

Gets or sets the internal DTD subset.

This property, along with

System.Xml.XmlParserContext.DocTypeName ,

System.Xml.XmlParserContext.PublicId , and

System.Xml.XmlParserContext.SystemId , provides all the document type declaration information. It is used to find and load the necessary DTD information so that you get all the default attributes and entities defined in the DTD.

NamespaceManager

WriteTo

**[C#] public XmlNamespaceManager NamespaceManager {get; set;}**

**[C++] public: \_\_property XmlNamespaceManager\***

**get\_NamespaceManager();public: \_\_property void**

**set\_NamespaceManager(XmlNamespaceManager\*);**

**[VB] Public Property NamespaceManager As XmlNamespaceManager**

**[JScript] public function get NamespaceManager() :**

**XmlNamespaceManager;public function set**

**NamespaceManager(XmlNamespaceManager);**

### ***Description***

**Gets or sets the System.Xml.XmlNamespaceManager .**

**The NamespaceManager defines the current namespace scope and provides methods for looking up namespace information.**

**NameTable**

**WriteTo**

**[C#] public XmlNameTable NameTable {get; set;}**

**[C++] public: \_\_property XmlNameTable\* get\_NameTable();public:**

**\_\_property void set\_NameTable(XmlNameTable\*);**

**[VB] Public Property NameTable As XmlNameTable**

**[JScript] public function get NameTable() : XmlNameTable;public function**

**set NameTable(XmlNameTable);**

### ***Description***

**Gets the System.Xml.XmlNameTable used to atomize strings.**

PublicId

WriteTo

**[C#] public string PublicId {get; set;}**

**[C++] public: \_\_property String\* get\_PublicId();public: \_\_property void**

**set\_PublicId(String\*);**

**[VB] Public Property PublicId As String**

**[JScript] public function get PublicId() : String;public function set**

**PublicId(String);**

### ***Description***

**Gets or sets the public identifier.**

**This property, along with**

**System.Xml.XmlParserContext.DocTypeName ,**

**System.Xml.XmlParserContext.SystemId , and**

**System.Xml.XmlParserContext.InternalSubset , provides all the document**

**type declaration information. It is used to find and load the required DTD**

**information so that you get all the default attributes and entities defined in**

**the DTD.**

SystemId

WriteTo

**[C#] public string SystemId {get; set;}**

**[C++] public: \_\_property String\* get\_SystemId();public: \_\_property void**

1 **set\_SystemId(String\*);**

2 **[VB] Public Property SystemId As String**

3 **[JScript] public function get SystemId() : String;public function set**

4 **SystemId(String);**

5  
6 ***Description***

7 **Gets or sets the system identifier.**

8 **This property, along with**

9 **System.Xml.XmlParserContext.DocTypeName ,**

10 **System.Xml.XmlParserContext.PublicId , and**

11 **System.Xml.XmlParserContext.InternalSubset , provides all the document**

12 **type declaration information. It is used to find and load the necessary DTD**

13 **information so that you get all the default attributes and entities defined in**

14 **the DTD.**

15 **XmlLang**

16 **WriteTo**

17  
18 **[C#] public string XmlLang {get; set;}**

19 **[C++] public: \_\_property String\* get\_XmlLang();public: \_\_property void**

20 **set\_XmlLang(String\*);**

21 **[VB] Public Property XmlLang As String**

22 **[JScript] public function get XmlLang() : String;public function set**

23 **XmlLang(String);**

24  
25 ***Description***

**Gets or sets the current xml:lang scope.**

**For details on valid xml:lang values, refer to section 2.12 of the W3C**

**Extensible Markup Language (XML) 1.0 recommendation.**

XmlSpace

WriteTo

**[C#] public XmlSpace XmlSpace {get; set;}**

**[C++] public: \_\_property XmlSpace get \_XmlSpace();public: \_\_property void  
set \_XmlSpace(XmlSpace);**

**[VB] Public Property XmlSpace As XmlSpace**

**[JScript] public function get XmlSpace() : XmlSpace;public function set  
XmlSpace(XmlSpace);**

### ***Description***

**Gets or sets the current xml:space scope.**

XmlProcessingInstruction class (System.Xml)

ToString

### ***Description***

**Represents a processing instruction, which XML defines to keep  
processor-specific information in the text of the document.**

XmlProcessingInstruction

***Example Syntax:***

ToString

1  
2 **[C#] protected internal XmlProcessingInstruction(string target, string data,**  
3 **XmlDocument doc);**

4 **[C++] internal: XmlProcessingInstruction(String\* target, String\* data,**  
5 **XmlDocument\* doc);**

6 **[VB] Protected Friend Sub New(ByVal target As String, ByVal data As**  
7 **String, ByVal doc As XmlDocument)**

8 **[JScript] package function XmlProcessingInstruction(target : String, data :**  
9 **String, doc : XmlDocument);**

10  
11 ***Description***

12 Attributes

13 BaseURI

14 ChildNodes

15 Data

16 ToString

17  
18  
19 ***Description***

20 **Gets or sets the content of the processing instruction, excluding the**  
21 **target.**

22 FirstChild

23 HasChildNodes

24 InnerText

25 ToString

**Description**

**Gets or sets the concatenated values of the node and all its children.**

InnerXml

IsReadOnly

Item

Item

LastChild

LocalName

ToString

**Description**

**Gets the local name of the node.**

Name

ToString

**[C#] public override string Name {get;}**

**[C++] public: \_\_property virtual String\* get\_Name();**

**[VB] Overrides Public ReadOnly Property Name As String**

**[JScript] public function get Name() : String;**

**Description**

**Gets the qualified name of the node.**

NamespaceURI

NextSibling

NodeType

ToString

### **Description**

**Gets the type of the current node.**

OuterXml

OwnerDocument

ParentNode

Prefix

PreviousSibling

Target

ToString

### **Description**

**Gets the target of the processing instruction.**

Value

ToString

**[C#] public override string Value {get; set;}**

**[C++] public: \_\_property virtual String\* get\_Value();public: \_\_property**

**virtual void set\_Value(String\*);**



1 **[VB] Overrides Public Property Value As String**

2 **[JScript] public function get Value() : String;public function set**

3 **Value(String);**

4  
5 ***Description***

6 **Gets or sets the value of the node.**

7 **CloneNode**

8  
9 **[C#] public override XmlNode CloneNode(bool deep);**

10 **[C++] public: XmlNode\* CloneNode(bool deep);**

11 **[VB] Overrides Public Function CloneNode(ByVal deep As Boolean) As**

12 **XmlNode**

13 **[JScript] public override function CloneNode(deep : Boolean) : XmlNode;**

14  
15 ***Description***

16 **Creates a duplicate of this node.**

17 ***Return Value:* The duplicate node.**

18 **CloneNode serves as a copy constructor for nodes. For processing**  
19 **instruction nodes, the cloned node always includes the target and data. To see**  
20 **how this method behaves with other node types, see**  
21 **System.Xml.XmlNode.CloneNode(System.Boolean) method in the XmlNode**  
22 **class. true to recursively clone the subtree under the specified node; false to**  
23 **clone only the node itself.**

24 **WriteContentTo**

1  
2 **[C#] public override void WriteContentTo(XmlWriter w);**

3 **[C++] public: void WriteContentTo(XmlWriter\* w);**

4 **[VB] Overrides Public Sub WriteContentTo(ByVal w As XmlWriter)**

5 **[JScript] public override function WriteContentTo(w : XmlWriter);**

6  
7 ***Description***

8 **Saves all the children of the node to the specified**  
9 **System.Xml.XmlWriter . The XmlWriter to which you want to save.**

10 WriteTo

11  
12 **[C#] public override void WriteTo(XmlWriter w);**

13 **[C++] public: void WriteTo(XmlWriter\* w);**

14 **[VB] Overrides Public Sub WriteTo(ByVal w As XmlWriter)**

15 **[JScript] public override function WriteTo(w : XmlWriter);**

16  
17 ***Description***

18 **Saves the node to the specified System.Xml.XmlWriter . The**  
19 **XmlWriter to which you want to save.**

20 XmlQualifiedName class (System.Xml)

21 WriteTo

22  
23  
24 ***Description***

25 **Represents an XML qualified name.**

1       An XML qualified name is a namespace qualified local name, in the  
2 **format namespace:localname .**

3       WriteTo

4  
5 **[C#] public static readonly XmlQualifiedName Empty;**

6 **[C++] public: static XmlQualifiedName\* Empty;**

7 **[VB] Public Shared ReadOnly Empty As XmlQualifiedName**

8 **[JScript] public static var Empty : XmlQualifiedName;**

9  
10 ***Description***

11       Provides an empty XmlQualifiedName .

12       This field allows you to do the following: XmlQualifiedName Qname =  
13 **new XmlQualifiedName ("item","http://www.contoso.com/");**  
14 **Qname.IsEmpty; //Returns false. Qname.Empty; //Sets the namespace and**  
15 **local name to String.Empty. Qname.IsEmpty; //Returns true.**

16       XmlQualifiedName

17 ***Example Syntax:***

18       WriteTo

19  
20 **[C#] public XmlQualifiedName();**

21 **[C++] public: XmlQualifiedName();**

22 **[VB] Public Sub New()**

23 **[JScript] public function XmlQualifiedName(); Initializes a new instance of**  
24 **the XmlQualifedName class.**

## Description

Initializes a new instance of the XmlQualifiedName class.

The object created has no name or namespace defined.

XmlQualifiedName

### Example Syntax:

WriteTo

[C#] public XmlQualifiedName(string name);

[C++] public: XmlQualifiedName(String\* name);

[VB] Public Sub New(ByVal name As String)

[JScript] public function XmlQualifiedName(name : String);

## Description

Initializes a new instance of the XmlQualifiedName class with the specified name.

The object created does not have a namespace defined. The local name to use as the name of the XmlQualifiedName object.

XmlQualifiedName

### Example Syntax:

WriteTo

[C#] public XmlQualifiedName(string name, string ns);

[C++] public: XmlQualifiedName(String\* name, String\* ns);

[VB] Public Sub New(ByVal name As String, ByVal ns As String)

1 [JScript] public function XmlQualifiedName(name : String, ns : String);

2  
3 **Description**

4       Initializes a new instance of the XmlQualifedName class with the  
5 specified name and namespace. The local name to use as the name of the  
6 XmlQualifiedName object. The namespace for the XmlQualifiedName object.

7       IsEmpty

8       WriteTo

9  
10 [C#] public bool IsEmpty {get;}

11 [C++] public: \_\_property bool get\_IsEmpty();

12 [VB] Public ReadOnly Property IsEmpty As Boolean

13 [JScript] public function get IsEmpty() : Boolean;

14  
15 **Description**

16       Gets a value indicating whether the XmlQualifedName is empty.

17       Name

18       WriteTo

19  
20 [C#] public string Name {get;}

21 [C++] public: \_\_property String\* get\_Name();

22 [VB] Public ReadOnly Property Name As String

23 [JScript] public function get Name() : String;

24  
25 **Description**

1 Gets a string representation of the qualified name of the  
2 XmlQualifiedName .

3 Namespace

4 WriteTo

5  
6 [C#] public string Namespace {get;}

7 [C++] public: \_\_property String\* get\_Namespace();

8 [VB] Public ReadOnly Property Namespace As String

9 [JScript] public function get Namespace() : String;

10  
11 **Description**

12 Gets a string representation of the namespace of the  
13 XmlQualifiedName .

14 Equals

15  
16 [C#] public override bool Equals(object other);

17 [C++] public: bool Equals(Object\* other);

18 [VB] Overrides Public Function Equals(ByVal other As Object) As Boolean

19 [JScript] public override function Equals(other : Object) : Boolean;

20  
21 **Description**

22 Checks if the specified XmlQualifiedName is the same instance object  
23 as this object.

24 **Return Value:** true if the two are the same instance object; otherwise false .

25 The XmlQualifiedName to compare.

## GetHashCode

**[C#]** public override int GetHashCode();  
**[C++]** public: int GetHashCode();  
**[VB]** Overrides Public Function GetHashCode() As Integer  
**[JScript]** public override function GetHashCode() : int;

### *Description*

Returns the hash code for the XmlQualifiedName .

**Return Value:** A hash code for this object.

op\_Equality

**[C#]** public static bool operator ==(XmlQualifiedName a, XmlQualifiedName b);

**[C++]** public: static bool op\_Equality(XmlQualifiedName\* a, XmlQualifiedName\* b);

**[VB]** returnValue = XmlQualifiedName.op\_Equality(a, b)

**[JScript]** returnValue = a == b;

### *Description*

Compares two XmlQualifiedName objects.

**Return Value:** true if the two objects have the same name and namespace values; otherwise false . An XmlQualifiedName to compare. An XmlQualifiedName to compare.

op\_Inequality

1  
2 **[C#] public static bool operator !=(XmlQualifiedName a, XmlQualifiedName**  
3 **b);**

4 **[C++] public: static bool op\_Inequality(XmlQualifiedName\* a,**  
5 **XmlQualifiedName\* b);**

6 **[VB] returnValue = XmlQualifiedName.op\_Inequality(a, b)**

7 **[JScript] returnValue = a != b;**

8  
9 ***Description***

10 **Compares two XmlQualifiedName objects.**

11 ***Return Value:* true if the name and namespace values for the two objects**  
12 **differ; otherwise false . An XmlQualifiedName to compare. An**  
13 **XmlQualifiedName to compare.**

14 ***ToString***

15  
16 **[C#] public override string ToString();**

17 **[C++] public: String\* ToString();**

18 **[VB] Overrides Public Function ToString() As String**

19 **[JScript] public override function ToString() : String; Returns the string**  
20 **value of the XmlQualifiedName .**

21  
22 ***Description***

23 **Returns the string value of the XmlQualifiedName .**

24 ***Return Value:* The string value of the XmlQualifiedName in the format of**  
25 **.**



1 **namespace:localname** . If the object does not have a namespace defined, this  
2 **method returns just the local name.**

3 ToString

4  
5 **[C#] public static string ToString(string name, string ns);**

6 **[C++] public: static String\* ToString(String\* name, String\* ns);**

7 **[VB] Public Shared Function ToString(ByVal name As String, ByVal ns As  
8 String) As String**

9 **[JScript] public static function ToString(name : String, ns : String) : String;**

10  
11 ***Description***

12 **Returns the string value of the XmlQualifiedName .**

13 ***Return Value:* The string value of the XmlQualifiedName in the format of**  
14 **namespace:localname . If the object does not have a namespace defined, this**  
15 **method returns just the local name. The name of the object. The namespace**  
16 **of the object.**

17 XmlReader class (System.Xml)

18 ToString

19  
20  
21 ***Description***

22 **Represents a reader that provides fast, non-cached, forward-only**  
23 **access to XML data.**

24 **XmlReader provides forward-only, read-only access to a stream of**  
25 **XML data. The current node refers to the node on which the reader is**

1 positioned. The reader is advanced using any of the read methods and  
2 properties reflect the value of the current node.

3 XmlReader

4 *Example Syntax:*

5 ToString

6  
7 [C#] protected XmlReader();

8 [C++] protected: XmlReader();

9 [VB] Protected Sub New()

10 [JScript] protected function XmlReader();

11 AttributeCount

12 ToString

13  
14 [C#] public abstract int AttributeCount {get;}

15 [C++] public: \_\_property virtual int get\_AttributeCount() = 0;

16 [VB] MustOverride Public ReadOnly Property AttributeCount As Integer

17 [JScript] public abstract function get AttributeCount() : int;

18  
19 *Description*

20 When overridden in a derived class, gets the number of attributes on  
21 the current node.

22 This property is relevant to Element , DocumentType and  
23 XmlDeclaration nodes only. (Other node types do not have attributes.) See  
24 System.Xml.XmlTextReader.AttributeCount (in the XmlTextReader class)  
25 for an example using this method.

BaseURI

ToString

**[C#] public abstract string BaseURI {get;}**

**[C++] public: \_\_property virtual String\* get\_BaseURI() = 0;**

**[VB] MustOverride Public ReadOnly Property BaseURI As String**

**[JScript] public abstract function get BaseURI() : String;**

### ***Description***

**When overridden in a derived class, gets the base URI of the current node.**

**A networked XML document is comprised of chunks of data aggregated using various W3C standard inclusion mechanisms and therefore contains nodes that come from different places. DTD entities are an example of this, but this is not limited to DTDs. The base URI tells you where these nodes came from. If there is no base URI for the nodes being returned (for example, they were parsed from an in-memory string), String.Empty is returned.**

CanResolveEntity

ToString

**[C#] public virtual bool CanResolveEntity {get;}**

**[C++] public: \_\_property virtual bool get\_CanResolveEntity();**

**[VB] Overridable Public ReadOnly Property CanResolveEntity As Boolean**

**[JScript] public function get CanResolveEntity() : Boolean;**

## ***Description***

Gets a value indicating whether this reader can parse and resolve entities.

This property always returns false for implementations of `XmlReader` that do not support schema or DTD information. In this case, calling `System.Xml.XmlReader.ResolveEntity` throws an exception.

Depth

ToString

**[C#] public abstract int Depth {get;}**

**[C++] public: \_\_property virtual int get\_Depth() = 0;**

**[VB] MustOverride Public ReadOnly Property Depth As Integer**

**[JScript] public abstract function get Depth() : int;**

## ***Description***

When overridden in a derived class, gets the depth of the current node in the XML document.

EOF

ToString

**[C#] public abstract bool EOF {get;}**

**[C++] public: \_\_property virtual bool get\_EOF() = 0;**

**[VB] MustOverride Public ReadOnly Property EOF As Boolean**

**[JScript] public abstract function get EOF() : Boolean;**

## **Description**

When overridden in a derived class, gets a value indicating whether the reader is positioned at the end of the stream.

HasAttributes

ToString

[C#] public virtual bool HasAttributes {get;}

[C++] public: \_\_property virtual bool get\_HasAttributes();

[VB] Overridable Public ReadOnly Property HasAttributes As Boolean

[JScript] public function get HasAttributes() : Boolean;

## **Description**

Gets a value indicating whether the current node has any attributes.

HasValue

ToString

[C#] public abstract bool HasValue {get;}

[C++] public: \_\_property virtual bool get\_HasValue() = 0;

[VB] MustOverride Public ReadOnly Property HasValue As Boolean

[JScript] public abstract function get HasValue() : Boolean;

## **Description**

When overridden in a derived class, gets a value indicating whether the current node can have a System.Xml.XmlReader.Value .

The following table lists node types that have a value to return.

IsDefault

ToString

[C#] public abstract bool IsDefault {get;}

[C++] public: \_\_property virtual bool get\_IsDefault() = 0;

[VB] MustOverride Public ReadOnly Property IsDefault As Boolean

[JScript] public abstract function get IsDefault() : Boolean;

### *Description*

When overridden in a derived class, gets a value indicating whether the current node is an attribute that was generated from the default value defined in the DTD or schema.

IsDefault always returns false for implementations of XmlReader that do not support schema or DTD information. This property applies only to an attribute node.

IsEmptyElement

ToString

[C#] public abstract bool IsEmptyElement {get;}

[C++] public: \_\_property virtual bool get\_IsEmptyElement() = 0;

[VB] MustOverride Public ReadOnly Property IsEmptyElement As Boolean

[JScript] public abstract function get IsEmptyElement() : Boolean;

### *Description*

When overridden in a derived class, gets a value indicating whether the current node is an empty element (for example, ).

This property enables you to determine the difference between the following: ( IsEmptyElement is true ).

Item

ToString

[C#] public abstract string this[int i] {get;}

[C++] public: \_\_property virtual String\* get\_Item(int i) = 0;

[VB] MustOverride Public Default ReadOnly Property Item(ByVal i As Integer) As String

[JScript] abstract returnValue = XmlReaderObject.Item(i); When overridden in a derived class, gets the value of the attribute.

### *Description*

When overridden in a derived class, gets the value of the attribute with the specified index.

This property does not move the reader. The index of the attribute.

Item

ToString

[C#] public abstract string this[string name] {get;}

[C++] public: \_\_property virtual String\* get\_Item(String\* name) = 0;

[VB] MustOverride Public Default ReadOnly Property Item(ByVal name As String) As String

**[JScript] abstract returnValue = XmlReaderObject.Item(name);**

### ***Description***

When overridden in a derived class, gets the value of the attribute with the specified **System.Xml.XmlReader.Name** .

This property does not move the reader. The qualified name of the attribute.

Item

ToString

**[C#] public abstract string this[string name, string namespaceURI] {get;}**

**[C++] public: \_\_property virtual String\* get\_Item(String\* name, String\* namespaceURI) = 0;**

**[VB] MustOverride Public Default ReadOnly Property Item(ByVal name As String, ByVal namespaceURI As String) As String**

**[JScript] abstract returnValue = XmlReaderObject.Item(name, namespaceURI);**

### ***Description***

When overridden in a derived class, gets the value of the attribute with the specified **System.Xml.XmlReader.LocalName** and **System.Xml.XmlReader.NamespaceURI** .

This property does not move the reader. The local name of the attribute. The namespace URI of the attribute.

LocalName



ToString

**[C#] public abstract string LocalName {get;}**

**[C++] public: \_\_property virtual String\* get\_LocalName() = 0;**

**[VB] MustOverride Public ReadOnly Property LocalName As String**

**[JScript] public abstract function get LocalName() : String;**

### ***Description***

When overridden in a derived class, gets the local name of the current node.

Name

ToString

**[C#] public abstract string Name {get;}**

**[C++] public: \_\_property virtual String\* get\_Name() = 0;**

**[VB] MustOverride Public ReadOnly Property Name As String**

**[JScript] public abstract function get Name() : String;**

### ***Description***

When overridden in a derived class, gets the qualified name of the current node.

NamespaceURI

ToString

**[C#] public abstract string NamespaceURI {get;}**

1 **[C++] public: \_\_property virtual String\* get\_NamespaceURI() = 0;**

2 **[VB] MustOverride Public ReadOnly Property NamespaceURI As String**

3 **[JScript] public abstract function get NamespaceURI() : String;**

4  
5 ***Description***

6 **When overridden in a derived class, gets the namespace URI (as**  
7 **defined in the W3C Namespace specification) of the node on which the reader**  
8 **is positioned.**

9 **This property is relevant to Element and Attribute nodes only.**

10 **NameTable**

11 **ToString**

12  
13 **[C#] public abstract XmlNameTable NameTable {get;}**

14 **[C++] public: \_\_property virtual XmlNameTable\* get\_NameTable() = 0;**

15 **[VB] MustOverride Public ReadOnly Property NameTable As**

16 **XmlNameTable**

17 **[JScript] public abstract function get NameTable() : XmlNameTable;**

18  
19 ***Description***

20 **When overridden in a derived class, gets the**  
21 **System.Xml.XmlNameTable associated with this implementation.**

22 **All node and attribute names returned from this class are atomized**  
23 **using the NameTable . When the same name is returned multiple times (for**  
24 **example, Customer ), then the same String object will be returned for that**

**name. This makes it possible for you to write efficient code that does object comparisons on these strings instead of expensive string comparisons.**

NodeType

ToString

**[C#] public abstract XmlNodeType NodeType {get;}**

**[C++] public: \_\_property virtual XmlNodeType get\_NodeType() = 0;**

**[VB] MustOverride Public ReadOnly Property NodeType As XmlNodeType**

**[JScript] public abstract function get NodeType() : XmlNodeType;**

### ***Description***

**When overridden in a derived class, gets the type of the current node.**

Prefix

ToString

**[C#] public abstract string Prefix {get;}**

**[C++] public: \_\_property virtual String\* get\_Prefix() = 0;**

**[VB] MustOverride Public ReadOnly Property Prefix As String**

**[JScript] public abstract function get Prefix() : String;**

### ***Description***

**When overridden in a derived class, gets the namespace prefix associated with the current node.**

QuoteChar

ToString

**[C#] public abstract char QuoteChar {get;}**

**[C++] public: \_\_property virtual \_\_wchar\_t get\_QuoteChar() = 0;**

**[VB] MustOverride Public ReadOnly Property QuoteChar As Char**

**[JScript] public abstract function get QuoteChar() : Char;**

### ***Description***

When overridden in a derived class, gets the quotation mark character used to enclose the value of an attribute node.

This property applies only to an attribute node.

ReadState

ToString

**[C#] public abstract ReadState ReadState {get;}**

**[C++] public: \_\_property virtual ReadState get\_ReadState() = 0;**

**[VB] MustOverride Public ReadOnly Property ReadState As ReadState**

**[JScript] public abstract function get ReadState() : ReadState;**

### ***Description***

When overridden in a derived class, gets the state of the reader.

Value

ToString

**[C#] public abstract string Value {get;}**

**[C++] public: \_\_property virtual String\* get\_Value() = 0;**

1 **[VB] MustOverride Public ReadOnly Property Value As String**

2 **[JScript] public abstract function get Value() : String;**

3  
4 ***Description***

5 **When overridden in a derived class, gets the text value of the current**  
6 **node.**

7 XmlLang

8 ToString

9  
10 **[C#] public abstract string XmlLang {get;}**

11 **[C++] public: \_\_property virtual String\* get \_XmlLang() = 0;**

12 **[VB] MustOverride Public ReadOnly Property XmlLang As String**

13 **[JScript] public abstract function get XmlLang() : String;**

14  
15 ***Description***

16 **When overridden in a derived class, gets the current xml:lang scope.**

17 **This property represents the xml:lang scope within which the current**  
18 **node resides. For example, here is an XML fragment with xml:lang set to US**  
19 **English in the root element: Fred When the reader is positioned on the name**  
20 **element, you can use this property to find that it is in the scope of a US**  
21 **English xml:lang attribute.**

22 XmlSpace

23 ToString

24  
25 **[C#] public abstract XmlSpace XmlSpace {get;}**

**[C++] public: \_\_property virtual XmlSpace get\_XmlSpace() = 0;**

**[VB] MustOverride Public ReadOnly Property XmlSpace As XmlSpace**

**[JScript] public abstract function get XmlSpace() : XmlSpace;**

### ***Description***

**When overridden in a derived class, gets the current xml:space scope.**

**Close**

**[C#] public abstract void Close();**

**[C++] public: virtual void Close() = 0;**

**[VB] MustOverride Public Sub Close()**

**[JScript] public abstract function Close();**

### ***Description***

**When overridden in a derived class, changes the System.Xml.XmlReader.ReadState to Closed.**

**This method also releases any resources held during reading. If Close has already been called, no action is performed.**

**GetAttribute**

**[C#] public abstract string GetAttribute(int i);**

**[C++] public: virtual String\* GetAttribute(int i) = 0;**

**[VB] MustOverride Public Function GetAttribute(ByVal i As Integer) As String**

**[JScript] public abstract function GetAttribute(i : int) : String;**

## **Description**

When overridden in a derived class, gets the value of the attribute with the specified index.

**Return Value:** The value of the specified attribute. This method does not move the reader. The index of the attribute. The index is zero-based. (The first attribute has index 0.)

GetAttribute

[C#] public abstract string GetAttribute(string name);

[C++] public: virtual String\* GetAttribute(String\* name) = 0;

[VB] MustOverride Public Function GetAttribute(ByVal name As String) As String

[JScript] public abstract function GetAttribute(name : String) : String; When overridden in a derived class, gets the value of an attribute.

## **Description**

When overridden in a derived class, gets the value of the attribute with the specified System.Xml.XmlReader.Name .

**Return Value:** The value of the specified attribute. If the attribute is not found, String.Empty is returned.

This method does not move the reader. The qualified name of the attribute.

GetAttribute

**[C#] public abstract string GetAttribute(string name, string namespaceURI);**

**[C++] public: virtual String\* GetAttribute(String\* name, String\* namespaceURI) = 0;**

**[VB] MustOverride Public Function GetAttribute(ByVal name As String, ByVal namespaceURI As String) As String**

**[JScript] public abstract function GetAttribute(name : String, namespaceURI : String) : String;**

### ***Description***

When overridden in a derived class, gets the value of the attribute with the specified **System.Xml.XmlReader.LocalName** and **System.Xml.XmlReader.NamespaceURI**.

***Return Value:*** The value of the specified attribute. If the attribute is not found, **String.Empty** is returned. This method does not move the reader.

The following XML contains an attribute in a specific namespace: You can lookup the **dt:type** attribute using one argument (prefix and local name) or two arguments (local name and namespace URI):

**String dt = reader.GetAttribute("dt:type");**

**String dt2 = reader.GetAttribute("type","urn:datatypes");**

To lookup the **xmlns:dt** attribute, use one of the following arguments:

**String dt3 = reader.GetAttribute("xmlns:dt");**

**String dt4 = reader.GetAttribute("dt",http://www.w3.org/2000/xmlns/);**

You can also get this information using the **System.Xml.XmlReader.Prefix** property. The local name of the attribute. The namespace URI of the attribute.



IsName

[C#] public static bool IsName(string str);

[C++] public: static bool IsName(String\* str);

[VB] Public Shared Function IsName(ByVal str As String) As Boolean

[JScript] public static function IsName(str : String) : Boolean;

### **Description**

Gets a value indicating whether the string argument is a valid XML name.

**Return Value:** true if the name is valid; otherwise, false .

This method uses the W3C XML 1.0 Recommendation (<http://www.w3.org/TR/2000/REC-xml-20001006#NT-Name>) to determine whether the name is valid. The name to validate.

IsNameToken

[C#] public static bool IsNameToken(string str);

[C++] public: static bool IsNameToken(String\* str);

[VB] Public Shared Function IsNameToken(ByVal str As String) As Boolean

[JScript] public static function IsNameToken(str : String) : Boolean;

### **Description**

Gets a value indicating whether or not the string argument is a valid XML name token.

**Return Value:** true if it is a valid name token; otherwise false .

This method uses the W3C XML 1.0 Recommendation (<http://www.w3.org/TR/2000/REC-xml-20001006#NT-Name>) to determine whether the name token is valid. The name token to validate.

IsStartElement

[C#] public virtual bool IsStartElement();

[C++] public: virtual bool IsStartElement();

[VB] Overridable Public Function IsStartElement() As Boolean

[JScript] public function IsStartElement() : Boolean; Tests if the current content node is a start tag.

### *Description*

Calls `System.Xml.XmlReader.MoveToContent` and tests if the current content node is a start tag or empty element tag.

**Return Value:** true if `MoveToContent` finds a start tag or empty element tag; false if a node type other than `XmlNodeType.Element` was found.

This method skips whitespace, comments, and processing instructions until the reader is positioned on a content node. The method then tests if the current node is an element.

IsStartElement

[C#] public virtual bool IsStartElement(string name);

[C++] public: virtual bool IsStartElement(String\* name);

[VB] Overridable Public Function IsStartElement(ByVal name As String) As Boolean

1 [JScript] public function IsStartElement(name : String) : Boolean;

3 **Description**

4       Calls System.Xml.XmlReader.MoveToContent and tests if the current  
5 content node is a start tag or empty element tag and if the  
6 System.Xml.XmlReader.Name property of the element found matches the  
7 given argument.

8 **Return Value:** true if the resulting node is an element and the Name property  
9 matches the specified string.

10       This method skips whitespace, comments, and processing instructions  
11 until the reader is positioned on a content node. The method then tests if the  
12 current node is an element. The string matched against the Name property of  
13 the element found.

14       IsStartElement

16 [C#] public virtual bool IsStartElement(string localname, string ns);

17 [C++] public: virtual bool IsStartElement(String\* localname, String\* ns);

18 [VB] Overridable Public Function IsStartElement(ByVal localname As  
19 String, ByVal ns As String) As Boolean

20 [JScript] public function IsStartElement(localname : String, ns : String) :  
21 Boolean;

23 **Description**

24       Calls System.Xml.XmlReader.MoveToContent and tests if the current  
25 content node is a start tag or empty element tag and if the

1 **System.Xml.XmlReader.LocalName** and

2 **System.Xml.XmlReader.NamespaceURI** properties of the element found  
3 match the given strings.

4 ***Return Value:*** true if the resulting node is an element.

5       This method skips whitespace, comments, and processing instructions  
6 until the reader is positioned on a content node. The method then tests if the  
7 current node is an element. The string to match against the LocalName  
8 property of the element found. The string to match against the  
9 NamespaceURI property of the element found.

10       LookupNamespace

11  
12 **[C#]** public abstract string LookupNamespace(string prefix);

13 **[C++]** public: virtual String\* LookupNamespace(String\* prefix) = 0;

14 **[VB]** MustOverride Public Function LookupNamespace(ByVal prefix As  
15 String) As String

16 **[JScript]** public abstract function LookupNamespace(prefix : String) : String;

17  
18 ***Description***

19       When overridden in a derived class, resolves a namespace prefix in the  
20 current element's scope.

21 ***Return Value:*** The namespace URI to which the prefix maps or null if no  
22 matching prefix is found.

23       In the preceding XML, if the reader is positioned on the href attribute,  
24 the prefix a is resolved by calling reader.LookupNamesapce("a") . The  
25 returned string is urn:456 . The prefix whose namespace URI you want to

1 resolve. To match the default namespace, pass an empty string. This string  
2 does not have to be atomized.

3 MoveToAttribute

4  
5 [C#] public abstract void MoveToAttribute(int i);

6 [C++] public: virtual void MoveToAttribute(int i) = 0;

7 [VB] MustOverride Public Sub MoveToAttribute(ByVal i As Integer)

8 [JScript] public abstract function MoveToAttribute(i : int);

9  
10 **Description**

11 When overridden in a derived class, moves to the attribute with the  
12 specified index. The index of the attribute.

13 MoveToAttribute

14  
15 [C#] public abstract bool MoveToAttribute(string name);

16 [C++] public: virtual bool MoveToAttribute(String\* name) = 0;

17 [VB] MustOverride Public Function MoveToAttribute(ByVal name As  
18 String) As Boolean

19 [JScript] public abstract function MoveToAttribute(name : String) : Boolean;

20 When overridden in a derived class, moves to the specified attribute.

21  
22 **Description**

23 When overridden in a derived class, moves to the attribute with the  
24 specified System.Xml.XmlReader.Name .

**Return Value:** true if the attribute is found; otherwise, false . If false , the reader's position does not change.

After calling `MoveToAttribute` , the `System.Xml.XmlReader.Name` , `System.Xml.XmlReader.NamespaceURI` , and `System.Xml.XmlReader.Prefix` properties reflect the properties of that attribute. The qualified name of the attribute.

`MoveToAttribute`

**[C#]** public abstract bool MoveToAttribute(string name, string ns);

**[C++]** public: virtual bool MoveToAttribute(String\* name, String\* ns) = 0;

**[VB]** MustOverride Public Function MoveToAttribute(ByVal name As String, ByVal ns As String) As Boolean

**[JScript]** public abstract function MoveToAttribute(name : String, ns : String) : Boolean;

### *Description*

When overridden in a derived class, moves to the attribute with the specified `System.Xml.XmlReader.LocalName` and `System.Xml.XmlReader.NamespaceURI` .

**Return Value:** true if the attribute is found; otherwise, false . If false , the reader's position does not change.

After calling `MoveToAttribute` , the `System.Xml.XmlReader.Name` , `System.Xml.XmlReader.NamespaceURI` , and `System.Xml.XmlReader.Prefix` properties reflect the properties of that attribute. The local name of the attribute. The namespace URI of the attribute.

MoveToContent

[C#] public virtual XmlNodeType MoveToContent();  
[C++] public: virtual XmlNodeType MoveToContent();  
[VB] Overridable Public Function MoveToContent() As XmlNodeType  
[JScript] public function MoveToContent() : XmlNodeType;

**Description**

Checks whether the current node is a content (non-whitespace text, CDATA , Element , EndElement , EntityReference , or EndEntity ) node. If the node is not a content node, the reader skips ahead to the next content node or end of file. It skips over nodes of the following type:

ProcessingInstruction , DocumentType , Comment , Whitespace , or SignificantWhitespace .

**Return Value:** The System.Xml.XmlReader.NodeType of the current node found by the method or XmlNodeType.None if the reader has reached the end of the input stream.

If the current node is an attribute node, this method moves the reader back to the element that owns the attribute.

MoveToElement

[C#] public abstract bool MoveToElement();  
[C++] public: virtual bool MoveToElement() = 0;  
[VB] MustOverride Public Function MoveToElement() As Boolean  
[JScript] public abstract function MoveToElement() : Boolean;

## **Description**

When overridden in a derived class, moves to the element that contains the current attribute node.

**Return Value:** true if the reader is positioned on an attribute (the reader moves to the element that owns the attribute); false if the reader is not positioned on an attribute (the position of the reader does not change).

Use this method to return to an element after navigating through its attributes. This method moves the reader to one of the following node types: Element , DocumentType , or XmlDeclaration .

MoveToFirstAttribute

[C#] public abstract bool MoveToFirstAttribute();

[C++] public: virtual bool MoveToFirstAttribute() = 0;

[VB] MustOverride Public Function MoveToFirstAttribute() As Boolean

[JScript] public abstract function MoveToFirstAttribute() : Boolean;

## **Description**

When overridden in a derived class, moves to the first attribute.

**Return Value:** true if an attribute exists (the reader moves to the first attribute); otherwise, false (the position of the reader does not change).

MoveToNextAttribute

[C#] public abstract bool MoveToNextAttribute();

[C++] public: virtual bool MoveToNextAttribute() = 0;



1 [VB] MustOverride Public Function MoveToNextAttribute() As Boolean

2 [JScript] public abstract function MoveToNextAttribute() : Boolean;

3  
4 **Description**

5 When overridden in a derived class, moves to the next attribute.

6 **Return Value:** true if there is a next attribute; false if there are no more  
7 attributes.

8 If the current node is an element node, this method is equivalent to  
9 System.Xml.XmlReader.MoveToFirstAttribute . If MoveToNextAttribute  
10 returns true , the reader moves to the next attribute; otherwise, the position  
11 of the reader does not change.

12 Read

13  
14 [C#] public abstract bool Read();

15 [C++] public: virtual bool Read() = 0;

16 [VB] MustOverride Public Function Read() As Boolean

17 [JScript] public abstract function Read() : Boolean;

18  
19 **Description**

20 When overridden in a derived class, reads the next node from the  
21 stream.

22 **Return Value:** true if the next node was read successfully; false if there are no  
23 more nodes to read.

24 When an XmlReader is first created and initialized, there is no  
25 information available. You must call Read to read the first node.

## ReadAttributeValue

**[C#] public abstract bool ReadAttributeValue();**

**[C++] public: virtual bool ReadAttributeValue() = 0;**

**[VB] MustOverride Public Function ReadAttributeValue() As Boolean**

**[JScript] public abstract function ReadAttributeValue() : Boolean;**

### *Description*

When overridden in a derived class, parses the attribute value into one or more Text or EntityReference nodes.

**Return Value:** true if there are nodes to return.

Use this method after calling

**System.Xml.XmlReader.MoveToAttribute(System.String)** to read through the text or entity reference nodes that make up the attribute value. The **System.Xml.XmlReader.Depth** of the attribute value nodes is one plus the depth of the attribute node; it increments and decrements by one when you step into and out of general entity references.

## ReadElementString

**[C#] public virtual string ReadElementString();**

**[C++] public: virtual String\* ReadElementString();**

**[VB] Overridable Public Function ReadElementString() As String**

**[JScript] public function ReadElementString() : String;** This is a helper method for reading simple text-only elements.

1  
2 **Description**

3 Reads a text-only element.

4 **Return Value:** The text contained in the element that was read. An empty  
5 string if the element is empty ( or ).

6 This is a helper method for reading simple text-only elements. It calls  
7 **System.Xml.XmlReader.MoveToContent** to find the next content node and  
8 then parses its value as a simple string.

9 ReadElementString

10  
11 **[C#]** public virtual string ReadElementString(string name);

12 **[C++]** public: virtual String\* ReadElementString(String\* name);

13 **[VB]** Overridable Public Function ReadElementString(ByVal name As  
14 String) As String

15 **[JScript]** public function ReadElementString(name : String) : String;

16  
17 **Description**

18 Checks that the **System.Xml.XmlReader.Name** property of the element  
19 found matches the given string before reading a text-only element.

20 **Return Value:** The text contained in the element that was read. An empty  
21 string if the element is empty ( or ).

22 This is a helper method for reading simple text-only elements. It calls  
23 **System.Xml.XmlReader.MoveToContent** to find the next content node and  
24 then parses its value as a simple string. The name to check.

25 ReadElementString

```

1
2 [C#] public virtual string ReadElementString(string localname, string ns);
3 [C++] public: virtual String* ReadElementString(String* localname, String*
4 ns);
5 [VB] Overridable Public Function ReadElementString(ByVal localname As
6 String, ByVal ns As String) As String
7 [JScript] public function ReadElementString(localname : String, ns : String) :
8 String;
9

```

### *Description*

Checks that the `System.Xml.XmlReader.LocalName` and `System.Xml.XmlReader.NamespaceURI` properties of the element found matches the given strings before reading a text-only element.

**Return Value:** The text contained in the element that was read. An empty string if the element is empty ( or ).

This is a helper method for reading simple text-only elements. It calls `System.Xml.XmlReader.MoveToContent` to find the next content node and then parses its value as a simple string. The local name to check. The namespace URI to check.

ReadEndElement

```

21
22 [C#] public virtual void ReadEndElement();
23 [C++] public: virtual void ReadEndElement();
24 [VB] Overridable Public Sub ReadEndElement()
25 [JScript] public function ReadEndElement();

```

## **Description**

Checks that the current content node is an end tag and advances the reader to the next node.

ReadInnerXml

[C#] public abstract string ReadInnerXml();

[C++] public: virtual String\* ReadInnerXml() = 0;

[VB] MustOverride Public Function ReadInnerXml() As String

[JScript] public abstract function ReadInnerXml() : String;

## **Description**

When overridden in a derived class, reads all the content, including markup, as a string.

**Return Value:** All the XML content, including markup, in the current node. If the current node has no children, an empty string is returned.

This method returns all the content of the current node including the markup. The current node (start tag) and corresponding end node (end tag) are not returned. For example, if you had the following: this ReadInnerXml returns this This method handles element and attribute nodes in the following way: Node Type Child Content Return Value Position After the Call Element text text After the end tag.

ReadOuterXml

[C#] public abstract string ReadOuterXml();

1 [C++] public: virtual String\* ReadOuterXml() = 0;

2 [VB] MustOverride Public Function ReadOuterXml() As String

3 [JScript] public abstract function ReadOuterXml() : String;

4  
5 **Description**

6 When overridden in a derived class, reads the content, including  
7 markup, representing this node and all its children.

8 **Return Value:** If the reader is positioned on an element or an attribute node,  
9 this method returns all the XML content, including markup, of the current  
10 node and all its children; otherwise, it returns an empty string.

11 This method is similar to System.Xml.XmlReader.ReadInnerXml  
12 except it also returns the start and end tags.

13 ReadStartElement

14  
15 [C#] public virtual void ReadStartElement();

16 [C++] public: virtual void ReadStartElement();

17 [VB] Overridable Public Sub ReadStartElement()

18 [JScript] public function ReadStartElement(); Checks that the current node  
19 is an element and advances the reader to the next node.

20  
21 **Description**

22 Checks that the current node is an element and advances the reader to  
23 the next node.

This method calls **System.Xml.XmlReader.IsStartElement** followed by **System.Xml.XmlReader.Read** to position you on the content of that element found in the input stream.

**ReadStartElement**

**[C#] public virtual void ReadStartElement(string name);**

**[C++] public: virtual void ReadStartElement(String\* name);**

**[VB] Overridable Public Sub ReadStartElement(ByVal name As String)**

**[JScript] public function ReadStartElement(name : String);**

### ***Description***

Checks that the current content node is an element with the given **System.Xml.XmlReader.Name** and advances the reader to the next node.

A call to this method corresponds to a call to **System.Xml.XmlReader.IsStartElement** followed by a call to **System.Xml.XmlReader.Read**. The qualified name of the element.

**ReadStartElement**

**[C#] public virtual void ReadStartElement(string localname, string ns);**

**[C++] public: virtual void ReadStartElement(String\* localname, String\* ns);**

**[VB] Overridable Public Sub ReadStartElement(ByVal localname As String, ByVal ns As String)**

**[JScript] public function ReadStartElement(localname : String, ns : String);**

### ***Description***

Checks that the current content node is an element with the given `System.Xml.XmlReader.LocalName` and `System.Xml.XmlReader.NamespaceURI` and advances the reader to the next node.

A call to this method corresponds to a call to `System.Xml.XmlReader.IsStartElement` followed by a call to `System.Xml.XmlReader.Read`. The local name of the element. The namespace URI of the element.

`ReadString`

[C#] `public abstract string ReadString();`

[C++] `public: virtual String* ReadString() = 0;`

[VB] `MustOverride Public Function ReadString() As String`

[JScript] `public abstract function ReadString() : String;`

### *Description*

When overridden in a derived class, reads the contents of an element or text node as a string.

**Return Value:** The contents of the element or text node. This can be an empty string if the reader is positioned on something other than an element or text node, or if there is no more text content to return in the current context.

If positioned on an element, `ReadString` concatenates all text, significant whitespace, whitespace, and CDATA section nodes together and returns the concatenated data as the element content. It stops when any



markup is encountered. This could occur in a mixed content model, or when an element end tag is read.

ResolveEntity

[C#] public abstract void ResolveEntity();

[C++] public: virtual void ResolveEntity() = 0;

[VB] MustOverride Public Sub ResolveEntity()

[JScript] public abstract function ResolveEntity();

### *Description*

When overridden in a derived class, resolves the entity reference for EntityReference nodes.

If the reader is positioned on an EntityReference node ( XmlNodeType.EntityReference ), if System.Xml.XmlReader.Read is called after calling this method, the entity replacement text is parsed. When the entity replacement text is finished, an EndEntity node is returned to close the entity reference scope.

Skip

[C#] public virtual void Skip();

[C++] public: virtual void Skip();

[VB] Overridable Public Sub Skip()

[JScript] public function Skip();

### *Description*

1       **Skips the current element.**

2       **In the following XML input if the reader is positioned on the node or**  
3 **any of its attributes, calling Skip positions the reader to the node.**

4       XmlResolver class (System.Xml)

5       ToString

6  
7  
8       ***Description***

9       **Resolves external XML resources named by a URI. This class is**  
10 **abstract .**

11       **XmlResolver is used to resolve external XML resources such as**  
12 **entities, DTDs or schemas. It is also used to process include and import**  
13 **elements found in Extensible StyleSheet Language (XSL) stylesheets or XML**  
14 **Schema Definition language (XSD) schemas.**

15       XmlResolver

16       ***Example Syntax:***

17       ToString

18  
19       **[C#] protected XmlResolver();**

20       **[C++] protected: XmlResolver();**

21       **[VB] Protected Sub New()**

22       **[JScript] protected function XmlResolver();**

23       Credentials

24       ToString

1  
2 **[C#] ICredentials Credentials {set;}**

3 **[C++] public: \_\_property virtual void set\_Credentials(ICredentials\*) = 0;**

4 **[VB] Property Credentials As ICredentials**

5 **[JScript] public abstract function set Credentials(ICredentials);**

6  
7 ***Description***

8 **When overridden in a derived class, sets the credentials used to**  
9 **authenticate Web requests.**

10 **If the virtual directory is configured to allow anonymous access, this**  
11 **property does not need to be set. Otherwise, the credentials of the user must**  
12 **be supplied. If credentials are needed but not supplied, the resolver uses**  
13 **default credentials (CredentialCache.DefaultCredential).**

14 **GetEntity**

15  
16 **[C#] public abstract object GetEntity(Uri absoluteUri, string role, Type**  
17 **ofObjectToReturn);**

18 **[C++] public: virtual Object\* GetEntity(Uri\* absoluteUri, String\* role,**  
19 **Type\* ofObjectToReturn) = 0;**

20 **[VB] MustOverride Public Function GetEntity(ByVal absoluteUri As Uri,**  
21 **ByVal role As String, ByVal ofObjectToReturn As Type) As Object**

22 **[JScript] public abstract function GetEntity(absoluteUri : Uri, role : String,**  
23 **ofObjectToReturn : Type) : Object;**

24  
25 ***Description***

When overridden in a derived class, maps a URI to an object containing the actual resource.

**Return Value:** A System.IO.Stream object or null if a type other than stream is specified.

This method is used when the caller wants to map a given URI into the object containing the actual resource that the URI represents. The type of object returned is negotiable however the implementation must always support System.IO.Stream . The URI returned from System.Xml.XmlResolver.ResolveUri(System.Uri, System.String) The current version does not use this parameter when resolving URIs. This is provided for future extensibility purposes. For example, this can be mapped to the xlink:role and used as an implementation specific argument in other scenarios. The type of object to return. The current version only returns System.IO.Stream objects.

#### ResolveUri

[C#] public abstract Uri ResolveUri(Uri baseUri, string relativeUri);

[C++] public: virtual Uri\* ResolveUri(Uri\* baseUri, String\* relativeUri) = 0;

[VB] MustOverride Public Function ResolveUri(ByVal baseUri As Uri, ByVal relativeUri As String) As Uri

[JScript] public abstract function ResolveUri(baseUri : Uri, relativeUri : String) : Uri;

#### Description

When overridden in a derived class, resolves the absolute URI from the base and relative URIs.

**Return Value:** The absolute URI or null if the relative URI can not be resolved.

The absolute URI may be used as the base URI for any subsequent requests for entities that are relative to this URI. The base URI used to resolve the relative URI The URI to resolve. The URI can be absolute or relative. If absolute, this value effectively replaces the *baseUri* value. If relative, it combines with the *baseUri* to make an absolute URI.

XmlSignificantWhitespace class (System.Xml)

ToString

### **Description**

Represents whitespace between markup in a mixed content mode or whitespace withing an xml:space= 'preserve' scope. This is also referred to as significant whitespace. These nodes are created automatically at System.Xml.XmlDocument.Load(System.String) time only if the System.Xml.XmlDocument.PreserveWhitespace flag is true .

XmlSignificantWhitespace

### **Example Syntax:**

ToString

[C#] protected internal XmlSignificantWhitespace(string strData, XmlDocument doc);

```

1  [C++] internal: XmlSignificantWhitespace(String* strData, XmlDocument*
2  doc);
3  [VB] Protected Friend Sub New(ByVal strData As String, ByVal doc As
4  XmlDocument)
5  [JScript] package function XmlSignificantWhitespace(strData : String, doc :
6  XmlDocument);

```

**Description**

- Attributes
- BaseURI
- ChildNodes
- Data
- FirstChild
- HasChildNodes
- InnerText
- InnerXml
- IsReadOnly
- Item
- Item
- LastChild
- Length
- LocalName
- ToString

1  
2  
3 **Description**

4 **Gets the local name of the node.**

5 Name

6 ToString

7  
8 **[C#] public override string Name {get;}**

9 **[C++] public: \_\_property virtual String\* get\_Name();**

10 **[VB] Overrides Public ReadOnly Property Name As String**

11 **[JScript] public function get Name() : String;**

12  
13 **Description**

14 **Gets the qualified name of the node.**

15 NamespaceURI

16 NextSibling

17 NodeType

18 ToString

19  
20  
21 **Description**

22 **Gets the type of the current node.**

23 OuterXml

24 OwnerDocument

25 ParentNode

Prefix

PreviousSibling

Value

ToString

### **Description**

**Gets or sets the value of the node.**

CloneNode

**[C#] public override XmlNode CloneNode(bool deep);**

**[C++] public: XmlNode\* CloneNode(bool deep);**

**[VB] Overrides Public Function CloneNode(ByVal deep As Boolean) As**

**XmlNode**

**[JScript] public override function CloneNode(deep : Boolean) : XmlNode;**

### **Description**

**Creates a duplicate of this node.**

**Return Value:** The duplicate node.

**This method serves as a generic copy constructor for nodes. The duplicate node has no parent (ParentNode returns null). true to recursively clone the subtree under the specified node; false to clone only the node itself (and its attributes if the node is an XmlElement).**

WriteContentTo



1  
2 **[C#] public override void WriteContentTo(XmlWriter w);**

3 **[C++] public: void WriteContentTo(XmlWriter\* w);**

4 **[VB] Overrides Public Sub WriteContentTo(ByVal w As XmlWriter)**

5 **[JScript] public override function WriteContentTo(w : XmlWriter);**

6  
7 ***Description***

8 **Saves all the children of the node to the specified XmlWriter. The**  
9 **XmlWriter where you want to save the current node.**

10 WriteTo

11  
12 **[C#] public override void WriteTo(XmlWriter w);**

13 **[C++] public: void WriteTo(XmlWriter\* w);**

14 **[VB] Overrides Public Sub WriteTo(ByVal w As XmlWriter)**

15 **[JScript] public override function WriteTo(w : XmlWriter);**

16  
17 ***Description***

18 **Saves the node to the specified XmlWriter. The XmlWriter where you**  
19 **want to save the node.**

20 XmlSpace enumeration (System.Xml)

21 WriteTo

22  
23  
24 ***Description***

25 **Specifies the current xml:space scope.**

WriteTo

**[C#] public const XmlSpace Default;**

**[C++] public: const XmlSpace Default;**

**[VB] Public Const Default As XmlSpace**

**[JScript] public var Default : XmlSpace;**

***Description***

**The xml:space scope equals default .**

WriteTo

**[C#] public const XmlSpace None;**

**[C++] public: const XmlSpace None;**

**[VB] Public Const None As XmlSpace**

**[JScript] public var None : XmlSpace;**

***Description***

**No xml:space scope.**

WriteTo

**[C#] public const XmlSpace Preserve;**

**[C++] public: const XmlSpace Preserve;**

**[VB] Public Const Preserve As XmlSpace**

**[JScript] public var Preserve : XmlSpace;**

1  
2 **Description**

3       **The xml:space scope equals preserve .**

4       XmlText class (System.Xml)

5       ToString

6  
7  
8 **Description**

9       **Represents the text content of an element or attribute.**

10       XmlText

11       **Example Syntax:**

12       ToString

13  
14 **[C#] protected internal XmlText(string strData, XmlDocument doc);**

15 **[C++] internal: XmlText(String\* strData, XmlDocument\* doc);**

16 **[VB] Protected Friend Sub New(ByVal strData As String, ByVal doc As**  
17 **XmlDocument)**

18 **[JScript] package function XmlText(strData : String, doc : XmlDocument);**

19  
20 **Description**

21       Attributes

22       BaseURI

23       ChildNodes

24       Data

25       FirstChild

MSDN Library

HasChildNodes

InnerText

InnerXml

IsReadOnly

Item

Item

LastChild

Length

LocalName

ToString

### **Description**

**Gets the local name of the node.**

Name

ToString

**[C#] public override string Name {get;}**

**[C++] public: \_\_property virtual String\* get\_Name();**

**[VB] Overrides Public ReadOnly Property Name As String**

**[JScript] public function get Name() : String;**

### **Description**

**Gets the qualified name of the node.**

NamespaceURI

NextSibling

NodeType

ToString

**Description**

**Gets the type of the current node.**

OuterXml

OwnerDocument

ParentNode

Prefix

PreviousSibling

Value

ToString

**Description**

**Gets or sets the value of the node.**

CloneNode

**[C#] public override XmlNode CloneNode(bool deep);**

**[C++] public: XmlNode\* CloneNode(bool deep);**

**[VB] Overrides Public Function CloneNode(ByVal deep As Boolean) As**

**XmlNode**

**[JScript] public override function CloneNode(deep : Boolean) : XmlNode;**

## **Description**

**Creates a duplicate of this node.**

**Return Value:** The cloned node.

**CloneNode** serves as a copy constructor for nodes. For text nodes, the cloned node always includes the data value. To see how this method behaves with other node types, see **System.Xml.XmlNode.CloneNode(System.Boolean)** method in the **XmlNode** class. **true** to recursively clone the subtree under the specified node; **false** to clone only the node itself.

## **SplitText**

**[C#]** public virtual XmlText SplitText(int offset);

**[C++]** public: virtual XmlText\* SplitText(int offset);

**[VB]** Overridable Public Function SplitText(ByVal offset As Integer) As

**XmlText**

**[JScript]** public function SplitText(offset : int) : XmlText;

## **Description**

**Splits the node into two nodes at the specified offset, keeping both in the tree as siblings.**

**After SplitText is called, the original node contains all the content up to the offset point. A new node of the same type, contains all the content at and after the offset point, and is inserted as the next sibling of the original node.**

**When the offset is equal to the length of this node, the new node has no data.**

**The offset at which to split the node.**

WriteContentTo

[C#] public override void WriteContentTo(XmlWriter w);

[C++] public: void WriteContentTo(XmlWriter\* w);

[VB] Overrides Public Sub WriteContentTo(ByVal w As XmlWriter)

[JScript] public override function WriteContentTo(w : XmlWriter);

**Description**

Saves all the children of the node to the specified  
System.Xml.XmlWriter . The XmlWriter to which you want to save.

WriteTo

[C#] public override void WriteTo(XmlWriter w);

[C++] public: void WriteTo(XmlWriter\* w);

[VB] Overrides Public Sub WriteTo(ByVal w As XmlWriter)

[JScript] public override function WriteTo(w : XmlWriter);

**Description**

Saves the node to the specified System.Xml.XmlWriter . The  
XmlWriter to which you want to save.

XmlTextReader class (System.Xml)

WriteTo

**Description**

Represents a reader that provides fast, non-cached, forward-only access to XML data.

**XmlTextReader** provides forward-only, read-only access to a stream of XML data. The current node refers to the node on which the reader is positioned. The reader is advanced using any of the read methods and properties reflect the value of the current node.

XmlTextReader

*Example Syntax:*

WriteTo

[C#] protected XmlTextReader();

[C++] protected: XmlTextReader();

[VB] Protected Sub New()

[JScript] protected function XmlTextReader(); Initializes a new instance of the XmlTextReader .

### *Description*

Initializes a new instance of the XmlTextReader .

XmlTextReader

*Example Syntax:*

WriteTo

[C#] public XmlTextReader(Stream input);

[C++] public: XmlTextReader(Stream\* input);

[VB] Public Sub New(ByVal input As Stream)



**[JScript] public function XmlTextReader(input : Stream);** Initializes a new instance of the XmlTextReader .

### **Description**

Initializes a new instance of the XmlTextReader class with the specified stream.

The XmlTextReader decodes the stream by wrapping it in a System.IO.StreamReader and calling SwitchEncoding according to the rules for XML encoding. The stream containing the XML data to read.

XmlTextReader

### **Example Syntax:**

WriteTo

**[C#] public XmlTextReader(string url);**

**[C++] public: XmlTextReader(String\* url);**

**[VB] Public Sub New(ByVal url As String)**

**[JScript] public function XmlTextReader(url : String);**

### **Description**

Initializes a new instance of the XmlTextReader class with the specified file. The URL for the file containing the XML data.

XmlTextReader

### **Example Syntax:**

WriteTo

1  
2 [C#] public XmlTextReader(TextReader input);

3 [C++] public: XmlTextReader(TextReader\* input);

4 [VB] Public Sub New(ByVal input As TextReader)

5 [JScript] public function XmlTextReader(input : TextReader);

6  
7 **Description**

8       **Initializes a new instance of the XmlTextReader class with the specified**  
9 **System.IO.TextReader .**

10       **It is assumed that the TextReader is already set to the correct**  
11 **encoding. This is used by clients that have already read some things from the**  
12 **stream in a multi-part MIME scenario. The TextReader containing the XML**  
13 **data to read.**

14       XmlTextReader

15       **Example Syntax:**

16       WriteTo

17  
18 [C#] protected XmlTextReader(XmlNameTable nt);

19 [C++] protected: XmlTextReader(XmlNameTable\* nt);

20 [VB] Protected Sub New(ByVal nt As XmlNameTable)

21 [JScript] protected function XmlTextReader(nt : XmlNameTable); **Initializes**  
22 **a new instance of the XmlTextReader class.**

23  
24 **Description**

1       **Initializes a new instance of the XmlTextReader class with the specified**  
2 **XmlNameTable. The XmlNameTable to use.**

3       XmlTextReader

4       *Example Syntax:*

5       WriteTo

7       **[C#] public XmlTextReader(Stream input, XmlNameTable nt);**

8       **[C++] public: XmlTextReader(Stream\* input, XmlNameTable\* nt);**

9       **[VB] Public Sub New(ByVal input As Stream, ByVal nt As XmlNameTable)**

10       **[JScript] public function XmlTextReader(input : Stream, nt :**  
11 **XmlNameTable);**

13       **Description**

14       **Initializes a new instance of the XmlTextReader class with the specified**  
15 **stream and System.Xml.XmlNameTable .**

16       **The XmlTextReader decodes the stream by wrapping it in a**  
17 **System.IO.StreamReader and calling SwitchEncoding according to the rules**  
18 **for XML encoding. The stream containing the XML data to read. The**  
19 **XmlNameTable to use.**

20       XmlTextReader

21       *Example Syntax:*

22       WriteTo

24       **[C#] public XmlTextReader(string url, Stream input);**

25       **[C++] public: XmlTextReader(String\* url, Stream\* input);**

[VB] Public Sub New(ByVal url As String, ByVal input As Stream)  
[JScript] public function XmlTextReader(url : String, input : Stream);

#### **Description**

Initializes a new instance of the XmlTextReader class with the specified URL and stream. The URL to use for resolving external resources. The System.Xml.XmlTextReader.BaseURI is set to this value. The stream containing the XML data to read.

XmlTextReader

#### **Example Syntax:**

WriteTo

[C#] public XmlTextReader(string url, TextReader input);

[C++] public: XmlTextReader(String\* url, TextReader\* input);

[VB] Public Sub New(ByVal url As String, ByVal input As TextReader)

[JScript] public function XmlTextReader(url : String, input : TextReader);

#### **Description**

Initializes a new instance of the XmlTextReader class with the specified URL and System.IO.TextReader . The URL to use for resolving external resources. The System.Xml.XmlTextReader.BaseURI is set to this value. The TextReader containing the XML data to read.

XmlTextReader

#### **Example Syntax:**

WriteTo

```

1  [C#] public XmlTextReader(string url, XmlNameTable nt);
2
3  [C++] public: XmlTextReader(String* url, XmlNameTable* nt);
4
5  [VB] Public Sub New(ByVal url As String, ByVal nt As XmlNameTable)
6
7  [JScript] public function XmlTextReader(url : String, nt : XmlNameTable);
8

```

### *Description*

Initializes a new instance of the `XmlTextReader` class with the specified file and `System.Xml.XmlNameTable`. The URL for the file containing the XML data to read. The `XmlNameTable` to use.

`XmlTextReader`

### *Example Syntax:*

`WriteTo`

```

15 [C#] public XmlTextReader(TextReader input, XmlNameTable nt);
16
17 [C++] public: XmlTextReader(TextReader* input, XmlNameTable* nt);
18
19 [VB] Public Sub New(ByVal input As TextReader, ByVal nt As
20 XmlNameTable)
21
22 [JScript] public function XmlTextReader(input : TextReader, nt :
23 XmlNameTable);
24

```

### *Description*

Initializes a new instance of the `XmlTextReader` class with the specified `System.IO.TextReader` and `System.Xml.XmlNameTable`.

1       It is assumed that the `TextReader` is already set to the correct  
2       encoding. This is used by clients that have already read some things from the  
3       stream in a multi-part MIME scenario. The `TextReader` containing the XML  
4       data to read. The `XmlNameTable` to use.

5       `XmlTextReader`

6       *Example Syntax:*

7       `WriteTo`

9       [C#] `public XmlTextReader(Stream xmlFragment, XmlNodeType fragType,  
10       XmlParserContext context);`

11       [C++] `public: XmlTextReader(Stream* xmlFragment, XmlNodeType  
12       fragType, XmlParserContext* context);`

13       [VB] `Public Sub New(ByVal xmlFragment As Stream, ByVal fragType As  
14       XmlNodeType, ByVal context As XmlParserContext)`

15       [JScript] `public function XmlTextReader(xmlFragment : Stream, fragType :  
16       XmlNodeType, context : XmlParserContext);`

18       *Description*

19       Initializes a new instance of the `XmlTextReader` class with the specified  
20       values.

21       This constructor parses the given string as a fragment of XML. If the  
22       XML fragment is an element or attribute, you can bypass the root level rules  
23       for well-formed XML documents. The stream containing the XML fragment  
24       to parse. The `System.Xml.XmlNodeType` of the XML fragment. This also  
25       determines what the fragment can contain. (See table below.) The

1 **System.Xml.XmlParserContext** in which the *xmlFragment* is to be parsed.

2 **This includes the System.Xml.XmlNameTable to use, the namespace scope,**  
3 **the current xml:lang, and the xml:space scope.**

4 **XmlTextReader**

5 ***Example Syntax:***

6 **WriteTo**

7  
8 **[C#] public XmlTextReader(string url, Stream input, XmlNameTable nt);**

9 **[C++] public: XmlTextReader(String\* url, Stream\* input, XmlNameTable\***  
10 **nt);**

11 **[VB] Public Sub New(ByVal url As String, ByVal input As Stream, ByVal nt**  
12 **As XmlNameTable)**

13 **[JScript] public function XmlTextReader(url : String, input : Stream, nt :**  
14 **XmlNameTable);**

15  
16 ***Description***

17 **Initializes a new instance of the XmlTextReader class with the specified**  
18 **URL, stream and System.Xml.XmlNameTable . The URL to use for resolving**  
19 **external resources. The System.Xml.XmlTextReader.BaseURI is set to this**  
20 **value. The stream containing the XML data to read. The XmlNameTable to**  
21 **use.**

22 **XmlTextReader**

23 ***Example Syntax:***

24 **WriteTo**

```

1
2 [C#] public XmlTextReader(string url, TextReader input, XmlNameTable
3 nt);
4 [C++] public: XmlTextReader(String* url, TextReader* input,
5 XmlNameTable* nt);
6 [VB] Public Sub New(ByVal url As String, ByVal input As TextReader,
7 ByVal nt As XmlNameTable)
8 [JScript] public function XmlTextReader(url : String, input : TextReader, nt
9 : XmlNameTable);
10

```

### *Description*

Initializes a new instance of the `XmlTextReader` class with the specified URL, `System.IO.TextReader` and `System.Xml.XmlNameTable`. The URL to use for resolving external resources. The `System.Xml.XmlTextReader.BaseURI` is set to this value. The `TextReader` containing the XML data to read. The `XmlNameTable` to use.

`XmlTextReader`

### *Example Syntax:*

`WriteTo`

```

21 [C#] public XmlTextReader(string xmlFragment, XmlNodeType fragType,
22 XmlParserContext context);
23 [C++] public: XmlTextReader(String* xmlFragment, XmlNodeType
24 fragType, XmlParserContext* context);
25 [VB] Public Sub New(ByVal xmlFragment As String, ByVal fragType As

```



XmlNodeType, ByVal context As XmlParserContext)

[JScript] public function XmlTextReader(xmlFragment : String, fragType :  
XmlNodeType, context : XmlParserContext);

### *Description*

Initializes a new instance of the XmlTextReader class with the specified values.

This constructor parses the given string as a fragment of XML. If the XML fragment is an element or attribute, you can bypass the root level rules for well-formed XML documents. This constructor can handle strings returned from System.Xml.XmlTextReader.ReadInnerXml . The string containing the XML fragment to parse. The System.Xml.XmlNodeType of the XML fragment. This also determines what the fragment string can contain. (See table below.) The System.Xml.XmlParserContext in which the *xmlFragment* is to be parsed. This includes the System.Xml.XmlNameTable to use, the namespace scope, the current xml:lang, and the xml:space scope.

AttributeCount

WriteTo

[C#] public override int AttributeCount {get;}

[C++] public: \_\_property virtual int get\_AttributeCount();

[VB] Overrides Public ReadOnly Property AttributeCount As Integer

[JScript] public function get AttributeCount() : int;

### *Description*

Gets the number of attributes on the current node.

This property is relevant to Element , DocumentType and XmlDeclaration nodes only. (Other node types do not have attributes.) The following example displays all attributes on the current node.

BaseURI

WriteTo

[C#] public override string BaseURI {get;}

[C++] public: \_\_property virtual String\* get \_BaseURI();

[VB] Overrides Public ReadOnly Property BaseURI As String

[JScript] public function get BaseURI() : String;

### *Description*

Gets the base URI of the current node.

A networked XML document is comprised of chunks of data aggregated using various W3C standard inclusion mechanisms and therefore contains nodes that come from different places. DTD entities are an example of this, but this is not limited to DTDs. The base URI tells you where these nodes came from. If there is no base URI for the nodes being returned (for example, they were parsed from an in-memory string), String.Empty is returned.

CanResolveEntity

Depth

WriteTo

**Description**

**Gets the depth of the current node in the XML document.**

Encoding

WriteTo

**[C#] public Encoding Encoding {get;}**

**[C++] public: \_\_property Encoding\* get\_Encoding();**

**[VB] Public ReadOnly Property Encoding As Encoding**

**[JScript] public function get Encoding() : Encoding;**

**Description**

**Gets the encoding of the document.**

**All encoding standards supported by the underlying operating system are supported.**

EOF

WriteTo

**[C#] public override bool EOF {get;}**

**[C++] public: \_\_property virtual bool get\_EOF();**

**[VB] Overrides Public ReadOnly Property EOF As Boolean**

**[JScript] public function get EOF() : Boolean;**

**Description**

1       **Gets a value indicating whether the reader is positioned at the end of**  
2 **the stream.**

3       HasAttributes

4       HasValue

5       WriteTo

6  
7  
8       **Description**

9       **Gets a value indicating whether the current node can have a**  
10 **System.Xml.XmlTextReader.Value .**

11       **The following table lists node types that have a value to return.**

12       IsDefault

13       WriteTo

14  
15 **[C#] public override bool IsDefault {get;}**

16 **[C++] public: \_\_property virtual bool get\_IsDefault();**

17 **[VB] Overrides Public ReadOnly Property IsDefault As Boolean**

18 **[JScript] public function get IsDefault() : Boolean;**

19  
20       **Description**

21       **Gets a value indicating whether the current node is an attribute that**  
22 **was generated from the default value defined in the DTD or schema.**

23       **This property applies only to attribute nodes.**

24       IsEmptyElement

25       WriteTo

**[C#] public override bool IsEmptyElement {get;}**

**[C++] public: \_\_property virtual bool get\_IsEmptyElement();**

**[VB] Overrides Public ReadOnly Property IsEmptyElement As Boolean**

**[JScript] public function get IsEmptyElement() : Boolean;**

### ***Description***

**Gets a value indicating whether the current node is an empty element (for example, ).**

**This property enables you to determine the difference between the following: ( IsEmptyElement is true ).**

Item

WriteTo

**[C#] public override string this[int i] {get;}**

**[C++] public: \_\_property virtual String\* get\_Item(int i);**

**[VB] Overrides Public Default ReadOnly Property Item(ByVal i As Integer) As String**

**[JScript] returnValue = XmlTextReaderObject.Item(i); Gets the value of the attribute.**

### ***Description***

**Gets the value of the attribute with the specified index.**

**This property does not move the reader. The index of the attribute.**

Item

WriteTo

[C#] public override string this[string name] {get;}

[C++] public: \_\_property virtual String\* get\_Item(String\* name);

[VB] Overrides Public Default ReadOnly Property Item(ByVal name As String) As String

[JScript] returnValue = XmlTextReaderObject.Item(name);

### *Description*

Gets the value of the attribute with the specified name.

This property does not move the reader. The qualified name of the attribute.

Item

WriteTo

[C#] public override string this[string name, string namespaceURI] {get;}

[C++] public: \_\_property virtual String\* get\_Item(String\* name, String\* namespaceURI);

[VB] Overrides Public Default ReadOnly Property Item(ByVal name As String, ByVal namespaceURI As String) As String

[JScript] returnValue = XmlTextReaderObject.Item(name, namespaceURI);

### *Description*

Gets the value of the attribute with the specified local name and namespace URI.

1       **This property does not move the reader. The local name of the**  
2 **attribute. The namespace URI of the attribute.**

3       LineNumber

4       WriteTo

6 **[C#] public int LineNumber {get;}**

7 **[C++] public: \_\_property int get\_LineNumber();**

8 **[VB] Public ReadOnly Property LineNumber As Integer**

9 **[JScript] public function get LineNumber() : int;**

11 ***Description***

12       **Gets the current line number.**

13       **This property is most commonly used for error reporting, but can be**  
14 **called at any time. The starting value for this property is 1 .**

15       LinePosition

16       WriteTo

18 **[C#] public int LinePosition {get;}**

19 **[C++] public: \_\_property int get\_LinePosition();**

20 **[VB] Public ReadOnly Property LinePosition As Integer**

21 **[JScript] public function get LinePosition() : int;**

23 ***Description***

24       **Gets the current line position.**

1       **This property is most commonly used for error reporting, but can be**  
2 **called at any time. The property's starting value is 1 .**

3       LocalName

4       WriteTo

6 **[C#] public override string LocalName {get;}**

7 **[C++] public: \_\_property virtual String\* get\_LocalName();**

8 **[VB] Overrides Public ReadOnly Property LocalName As String**

9 **[JScript] public function get LocalName() : String;**

11 ***Description***

12       **Gets the local name of the current node.**

13       Name

14       WriteTo

16 **[C#] public override string Name {get;}**

17 **[C++] public: \_\_property virtual String\* get\_Name();**

18 **[VB] Overrides Public ReadOnly Property Name As String**

19 **[JScript] public function get Name() : String;**

21 ***Description***

22       **Gets the qualified name of the current node.**

23       Namespaces

24       WriteTo



1  
2 **[C#] public bool Namespaces {get; set;}**

3 **[C++] public: \_\_property bool get\_Namespaces();public: \_\_property void**  
4 **set\_Namespaces(bool);**

5 **[VB] Public Property Namespaces As Boolean**

6 **[JScript] public function get Namespaces() : Boolean;public function set**  
7 **Namespaces(Boolean);**

8  
9 ***Description***

10 **Gets or sets a value indicating whether to do namespace support.**

11 **This property cannot be set after a read operation has occurred.**

12 NamespaceURI

13 WriteTo

14  
15 **[C#] public override string NamespaceURI {get;}**

16 **[C++] public: \_\_property virtual String\* get\_NamespaceURI();**

17 **[VB] Overrides Public ReadOnly Property NamespaceURI As String**

18 **[JScript] public function get NamespaceURI() : String;**

19  
20 ***Description***

21 **Gets the namespace URI (as defined in the W3C Namespace**  
22 **specification) of the node on which the reader is positioned.**

23 **This property is relevant to Element and Attribute nodes only.**

24 NameTable

25 WriteTo

1  
2 **[C#] public override XmlNameTable NameTable {get;}**

3 **[C++] public: \_\_property virtual XmlNameTable\* get\_NameTable();**

4 **[VB] Overrides Public ReadOnly Property NameTable As XmlNameTable**

5 **[JScript] public function get NameTable() : XmlNameTable;**

6  
7 ***Description***

8 Gets the System.Xml.XmlNameTable associated with this  
9 implementation.

10 All node and attribute names returned from this class are atomized  
11 using the NameTable . When the same name is returned multiple times (for  
12 example, Customer ), then the same String object will be returned for that  
13 name. This makes it possible for you to write efficient code that does object  
14 comparisons on these strings instead of expensive string comparisons.

15 NodeType

16 WriteTo

17  
18 **[C#] public override XmlNodeType NodeType {get;}**

19 **[C++] public: \_\_property virtual XmlNodeType get\_NodeType();**

20 **[VB] Overrides Public ReadOnly Property NodeType As XmlNodeType**

21 **[JScript] public function get NodeType() : XmlNodeType;**

22  
23 ***Description***

24 Gets the type of the current node.

**This property never returns the following XmlNodeType types:**

**Document , DocumentFragment , Entity , EndEntity , or Notation .**

Normalization

WriteTo

**[C#] public bool Normalization {get; set;}**

**[C++] public: \_\_property bool get\_Normalization();public: \_\_property void  
set\_Normalization(bool);**

**[VB] Public Property Normalization As Boolean**

**[JScript] public function get Normalization() : Boolean;public function set  
Normalization(Boolean);**

### ***Description***

**Gets or sets a value indicating whether to normalize whitespace and  
attribute values.**

**This property can be changed at any time and takes affect on the next  
read operation.**

Prefix

WriteTo

**[C#] public override string Prefix {get;}**

**[C++] public: \_\_property virtual String\* get\_Prefix();**

**[VB] Overrides Public ReadOnly Property Prefix As String**

**[JScript] public function get Prefix() : String;**

1  
2 **Description**

3 Gets the namespace prefix associated with the current node.

4 QuoteChar

5 WriteTo

6  
7 [C#] public override char QuoteChar {get;}

8 [C++] public: \_\_property virtual \_\_wchar\_t get\_QuoteChar();

9 [VB] Overrides Public ReadOnly Property QuoteChar As Char

10 [JScript] public function get QuoteChar() : Char;

11  
12 **Description**

13 Gets the quotation mark character used to enclose the value of an  
14 attribute node.

15 This property applies only to an attribute node.

16 ReadState

17 WriteTo

18  
19 [C#] public override ReadState ReadState {get;}

20 [C++] public: \_\_property virtual ReadState get\_ReadState();

21 [VB] Overrides Public ReadOnly Property ReadState As ReadState

22 [JScript] public function get ReadState() : ReadState;

23  
24 **Description**

25 Gets the state of the reader.

Value

WriteTo

[C#] public override string Value {get;}

[C++] public: \_\_property virtual String\* get\_Value();

[VB] Overrides Public ReadOnly Property Value As String

[JScript] public function get Value() : String;

### *Description*

**Gets the text value of the current node.**

WhitespaceHandling

WriteTo

[C#] public WhitespaceHandling WhitespaceHandling {get; set;}

[C++] public: \_\_property WhitespaceHandling

get\_WhitespaceHandling();public: \_\_property void

set\_WhitespaceHandling(WhitespaceHandling);

[VB] Public Property WhitespaceHandling As WhitespaceHandling

[JScript] public function get WhitespaceHandling() :

WhitespaceHandling;public function set

WhitespaceHandling(WhitespaceHandling);

### *Description*

**Gets or sets a value that specifies how whitespace is handled.**

This property can be changed at any time and takes affect on the next read operation.

XmlLang

WriteTo

**[C#] public override string XmlLang {get;}**

**[C++] public: \_\_property virtual String\* get\_XmlLang();**

**[VB] Overrides Public ReadOnly Property XmlLang As String**

**[JScript] public function get XmlLang() : String;**

### ***Description***

**Gets the current xml:lang scope.**

**This property represents the xml:lang scope within which the current node resides. For example, here is an XML fragment with xml:lang set to US English in the root element: Fred When the reader is positioned on the name element, you can use this property to find that it is in the scope of a US English xml:lang attribute.**

XmlResolver

WriteTo

**[C#] XmlResolver XmlResolver {set;}**

**[C++] public: \_\_property void set\_XmlResolver(XmlResolver\*);**

**[VB] Property XmlResolver As XmlResolver**

**[JScript] public function set XmlResolver(XmlResolver);**

**Description**

Sets the `System.Xml.XmlResolver` used for resolving DTD references.

The reader uses `XmlResolver` to resolve the location of the file loaded into the reader and also to resolve DTD references. For example, if your XML included the DOCTYPE declaration, the reader resolves this external file and ensures that the DTD is well-formed. The reader does not use the DTD for validation.

`XmlSpace`

`WriteTo`

**[C#]** `public override XmlSpace XmlSpace {get;}`

**[C++]** `public: __property virtual XmlSpace get_XmlSpace();`

**[VB]** Overrides Public ReadOnly Property `XmlSpace` As `XmlSpace`

**[JScript]** `public function get XmlSpace() : XmlSpace;`

**Description**

Gets the current `xml:space` scope.

`Close`

**[C#]** `public override void Close();`

**[C++]** `public: void Close();`

**[VB]** Overrides Public Sub `Close()`

**[JScript]** `public override function Close();`

1  
2 **Description**

3 **Changes the System.Xml.XmlReader.ReadState to Closed.**

4 **This method also releases any resources held while reading. If this**  
5 **reader was constructed using a stream, this method also calls Close on the**  
6 **underlying stream.**

7 **GetAttribute**

8  
9 **[C#] public override string GetAttribute(int i);**

10 **[C++] public: String\* GetAttribute(int i);**

11 **[VB] Overrides Public Function GetAttribute(ByVal i As Integer) As String**

12 **[JScript] public override function GetAttribute(i : int) : String; Gets the**  
13 **value of an attribute.**

14  
15 **Description**

16 **Gets the value of the attribute with the specified index.**

17 **Return Value:** The value of the specified attribute.

18 **This method does not move the reader. The index of the attribute. The**  
19 **index is zero-based. (The first attribute has index 0.)**

20 **GetAttribute**

21  
22 **[C#] public override string GetAttribute(string name);**

23 **[C++] public: String\* GetAttribute(String\* name);**

24 **[VB] Overrides Public Function GetAttribute(ByVal name As String) As**  
25 **String**



1 [JScript] public override function GetAttribute(name : String) : String;

2  
3 *Description*

4 Gets the value of the attribute with the specified name.

5 *Return Value:* The value of the specified attribute. If the attribute is not  
6 found, String.Empty is returned.

7 This method does not move the reader. The qualified name of the  
8 attribute.

9 GetAttribute

10  
11 [C#] public override string GetAttribute(string localName, string  
12 namespaceURI);

13 [C++] public: String\* GetAttribute(String\* localName, String\*  
14 namespaceURI);

15 [VB] Overrides Public Function GetAttribute(ByVal localName As String,  
16 ByVal namespaceURI As String) As String

17 [JScript] public override function GetAttribute(localName : String,  
18 namespaceURI : String) : String;

19  
20 *Description*

21 Gets the value of the attribute with the specified local name and  
22 namespace URI.

23 *Return Value:* The value of the specified attribute. If the attribute is not  
24 found, String.Empty is returned. This method does not move the reader.

The following XML contains an attribute in a specific namespace: You can lookup the dt:type attribute using one argument (prefix and local name) or two arguments (local name and namespace URI): String dt = reader.GetAttribute("dt:type"); String dt2 = reader.GetAttribute("type","urn:datatypes"); To lookup the xmlns:dt attribute, use one of the following arguments: String dt3 = reader.GetAttribute("xmlns:dt"); String dt4 = reader.GetAttribute("dt",http://www.w3.org/2000/xmlns/); You can also get this information using the System.Xml.XmlTextReader.Prefix property. The local name of the attribute. The namespace URI of the attribute.

GetRemainder

[C#] public TextReader GetRemainder();

[C++] public: TextReader\* GetRemainder();

[VB] Public Function GetRemainder() As TextReader

[JScript] public function GetRemainder() : TextReader;

### *Description*

**Gets the remainder of the buffered XML.**

Because XmlTextReader does a buffered Read , it must be able to return the remainder of the unused buffer so that no data is lost. This allows protocols (such as multi-part MIME) to package XML in the same stream as other things.

LookupNamespace

1  
2 [C#] public override string LookupNamespace(string prefix);

3 [C++] public: String\* LookupNamespace(String\* prefix);

4 [VB] Overrides Public Function LookupNamespace(ByVal prefix As String)  
5 As String

6 [JScript] public override function LookupNamespace(prefix : String) :  
7 String;

8  
9 **Description**

10 Resolves a namespace prefix in the current element's scope.

11 **Return Value:** The namespace URI to which the prefix maps or null if no  
12 matching prefix is found.

13 In the preceding XML, if the reader is positioned on the href attribute,  
14 the prefix a is resolved by calling reader.LookupNamesapce("a") . The  
15 returned string is urn:456 . The prefix whose namespace URI you want to  
16 resolve. To match the default namespace, pass an empty string. This string  
17 does not have to be atomized.

18 MoveToAttribute

19  
20 [C#] public override void MoveToAttribute(int i);

21 [C++] public: void MoveToAttribute(int i);

22 [VB] Overrides Public Sub MoveToAttribute(ByVal i As Integer)

23 [JScript] public override function MoveToAttribute(i : int);

24  
25 **Description**

Moves to the attribute with the specified index. The index of the attribute.

MoveToAttribute

[C#] public override bool MoveToAttribute(string name);

[C++] public: bool MoveToAttribute(String\* name);

[VB] Overrides Public Function MoveToAttribute(ByVal name As String) As

Boolean

[JScript] public override function MoveToAttribute(name : String) :

Boolean; Moves to the specified attribute.

### *Description*

Moves to the attribute with the specified name.

**Return Value:** true if the attribute is found; otherwise, false . If false , the reader's position does not change.

After calling MoveToAttribute , the System.Xml.XmlTextReader.Name , System.Xml.XmlTextReader.NamespaceURI , and System.Xml.XmlTextReader.Prefix properties will reflect the properties of that attribute. The qualified name of the attribute.

MoveToAttribute

[C#] public override bool MoveToAttribute(string localName, string namespaceURI);

[C++] public: bool MoveToAttribute(String\* localName, String\*

1 namespaceURI);

2 [VB] Overrides Public Function MoveToAttribute(ByVal localName As  
3 String, ByVal namespaceURI As String) As Boolean

4 [JScript] public override function MoveToAttribute(localName : String,  
5 namespaceURI : String) : Boolean;

6  
7 **Description**

8 Moves to the attribute with the specified local name and namespace  
9 URI.

10 **Return Value:** true if the attribute is found; otherwise, false . If false , the  
11 reader's position does not change.

12 After calling MoveToAttribute , the  
13 System.Xml.XmlTextReader.Name ,  
14 System.Xml.XmlTextReader.NamespaceURI , and  
15 System.Xml.XmlTextReader.Prefix properties will reflect the properties of  
16 that attribute. The local name of the attribute. The namespace URI of the  
17 attribute.

18 MoveToElement

19  
20 [C#] public override bool MoveToElement();

21 [C++] public: bool MoveToElement();

22 [VB] Overrides Public Function MoveToElement() As Boolean

23 [JScript] public override function MoveToElement() : Boolean;

24  
25 **Description**

Moves to the element that contains the current attribute node.

**Return Value:** true if the reader is positioned on an attribute (the reader moves to the element that owns the attribute); false if the reader is not positioned on an attribute (the position of the reader does not change).

Use this method to return to an element after navigating through its attributes. This method moves the reader to one of the following node types:

**Element , DocumentType , or XmlDeclaration .**

MoveToFirstAttribute

**[C#] public override bool MoveToFirstAttribute();**

**[C++] public: bool MoveToFirstAttribute();**

**[VB] Overrides Public Function MoveToFirstAttribute() As Boolean**

**[JScript] public override function MoveToFirstAttribute() : Boolean;**

### **Description**

Moves to the first attribute.

**Return Value:** true if an attribute exists (the reader moves to the first attribute); otherwise, false (the position of the reader does not change).

MoveToNextAttribute

**[C#] public override bool MoveToNextAttribute();**

**[C++] public: bool MoveToNextAttribute();**

**[VB] Overrides Public Function MoveToNextAttribute() As Boolean**

**[JScript] public override function MoveToNextAttribute() : Boolean;**

## ***Description***

Moves to the next attribute.

***Return Value:*** true if there is a next attribute; false if there are no more attributes.

If the current node is an element node, this method is equivalent to `System.Xml.XmlTextReader.MoveToFirstAttribute` . If `MoveToNextAttribute` returns true , the reader moves to the next attribute; otherwise, the position of the reader does not change.

Read

**[C#] public override bool Read();**

**[C++] public: bool Read();**

**[VB] Overrides Public Function Read() As Boolean**

**[JScript] public override function Read() : Boolean;**

## ***Description***

Reads the next node from the stream.

***Return Value:*** true if the next node was read successfully; false if there are no more nodes to read.

When a reader is first created and initialized, there is no information available. You must call `Read` to read the first node.

ReadAttributeValue

**[C#] public override bool ReadAttributeValue();**

**[C++] public: bool ReadAttributeValue();**

**[VB] Overrides Public Function ReadAttributeValue() As Boolean**

**[JScript] public override function ReadAttributeValue() : Boolean;**

### **Description**

Parses the attribute value into one or more Text or EntityReference nodes.

**Return Value:** true if there are nodes to return.

Use this method after calling MoveToAttribute to read through the text or entity reference nodes that make up the attribute value. The System.Xml.XmlReader.Depth of the attribute value nodes is one plus the depth of the attribute node; it increments and decrements by one when you step into and out of general entity references.

ReadBase64

**[C#] public int ReadBase64(byte[] array, int offset, int len);**

**[C++] public: int ReadBase64(unsigned char array \_\_gc[], int offset, int len);**

**[VB] Public Function ReadBase64(ByVal array) As Byte, ByVal offset As Integer, ByVal len As Integer) As Integer**

**[JScript] public function ReadBase64(array : Byte[], offset : int, len : int) : int;**

### **Description**

Decodes Base64 and returns the decoded binary bytes.

**Return Value:** The number of bytes written to the buffer.



1       **Like**

2       **System.Xml.XmlTextReader.ReadChars(System.Char[],System.Int32,System**  
3       **.Int32) , this method can be called successively to read large streams of**  
4       **embedded text. It decodes Base64 content and returns the decoded binary**  
5       **bytes (for example, an inline Base64 encoded GIF image) into the buffer. See**  
6       **RFC 1521. (You can obtain RFCs from the Request for Comments Web site**  
7       **at <http://www.rfc-editor.org>) The following example reads a file containing**  
8       **Base64 and BinHex data. The array of characters that serves as the buffer to**  
9       **which the text contents are written. The zero-based index into the array**  
10       **specifying where the method should begin to write to the buffer. The number**  
11       **of bytes to write into the buffer.**

12       **ReadBinHex**

13  
14       **[C#] public int ReadBinHex(byte[] array, int offset, int len);**

15       **[C++] public: int ReadBinHex(unsigned char array \_\_gc[], int offset, int len);**

16       **[VB] Public Function ReadBinHex(ByVal array() As Byte, ByVal offset As**  
17       **Integer, ByVal len As Integer) As Integer**

18       **[JScript] public function ReadBinHex(array : Byte[], offset : int, len : int) :**  
19       **int;**

20  
21       ***Description***

22       **Decodes BinHex and returns the decoded binary bytes.**

23       ***Return Value:* The number of bytes written to your buffer.**

24       **Like**

25       **System.Xml.XmlTextReader.ReadChars(System.Char[],System.Int32,System**

1 .Int32) , this method can be called successively to read large streams of  
2 embedded text. It decodes BinHex content and returns the decoded binary  
3 bytes (for example, an inline BinHex encoded GIF image) into the buffer. The  
4 byte array that serves as the buffer to which the decoded binary bytes are  
5 written. The zero-based index into the array specifying where the method  
6 should begin to write to the buffer. The number of bytes to write into the  
7 buffer.

8       ReadChars

9  
10 [C#] public int ReadChars(char[] buffer, int index, int count);

11 [C++] public: int ReadChars(\_\_wchar\_t buffer \_\_gc[], int index, int count);

12 [VB] Public Function ReadChars(ByVal buffer() As Char, ByVal index As  
13 Integer, ByVal count As Integer) As Integer

14 [JavaScript] public function ReadChars(buffer : Char[], index : int, count : int) :  
15 int;

16  
17 *Description*

18       Reads the text contents of an element into a character buffer. This  
19 method is designed to read large streams of embedded text by calling it  
20 successively.

21 *Return Value:* The number of characters read. This can be 0 if the reader is  
22 not positioned on an element or if there is no more text content to return in  
23 the current context.

24       This is the most efficient way to process very large streams of text  
25 embedded in an XML document. Rather than allocating large string objects,

1 ReadChars returns text content a buffer at a time. This method is designed to  
2 work only on element nodes. Other node types cause ReadChars to return 0.  
3 The array of characters that serves as the buffer to which the text contents  
4 are written. The position within *buffer* where the method should begin  
5 writing text contents. The number of characters to write into *buffer*.

6 ReadInnerXml

7  
8 [C#] public override string ReadInnerXml();

9 [C++] public: String\* ReadInnerXml();

10 [VB] Overrides Public Function ReadInnerXml() As String

11 [JScript] public override function ReadInnerXml() : String;

12  
13 **Description**

14 Reads all the content, including markup, as a string.

15 **Return Value:** All the XML content, including markup, in the current node. If  
16 the current node has no children, an empty string is returned.

17 This method returns all the content of the current node including the  
18 markup. The current node (start tag) and corresponding end node (end tag)  
19 are not returned. For example, if you had the following: this ReadInnerXml  
20 returns this This method handles element and attribute nodes in the following  
21 way: Node Type Child Content Return Value Position After the Call Element  
22 text text After the end tag.

23 ReadOuterXml

24  
25 [C#] public override string ReadOuterXml();

1 [C++] public: String\* ReadOuterXml();

2 [VB] Overrides Public Function ReadOuterXml() As String

3 [JScript] public override function ReadOuterXml() : String;

4  
5 **Description**

6 Reads the content, including markup, representing this node and all its  
7 children.

8 **Return Value:** If the reader is positioned on an element or an attribute node,  
9 this method returns all the XML content, including markup, of the current  
10 node and all its children; otherwise, it returns an empty string.

11 This method is similar to System.Xml.XmlTextReader.ReadInnerXml  
12 except it also returns the start and end tags.

13 ReadString

14  
15 [C#] public override string ReadString();

16 [C++] public: String\* ReadString();

17 [VB] Overrides Public Function ReadString() As String

18 [JScript] public override function ReadString() : String;

19  
20 **Description**

21 Reads the contents of an element or a text node as a string.

22 **Return Value:** The contents of the element or text node. This can be an empty  
23 string if the reader is positioned on something other than an element or text  
24 node, or if there is no more text content to return in the current context.

If positioned on an element, `ReadString` concatenates all text, significant whitespace, whitespace and `CData` section node types together and returns the concatenated data as the element content. It stops when any markup is encountered. This could occur in a mixed content model, or when an element end tag is read.

`ResolveEntity`

[C#] `public override void ResolveEntity();`

[C++] `public: void ResolveEntity();`

[VB] `Overrides Public Sub ResolveEntity()`

[JScript] `public override function ResolveEntity();`

### *Description*

Resolves the entity reference for `EntityReference` nodes.

`XmlTextReader` cannot resolve general entities. Calling this method will throw an exception.

`IXmlLineInfo.HasLineInfo`

[C#] `bool IXmlLineInfo.HasLineInfo();`

[C++] `bool IXmlLineInfo::HasLineInfo();`

[VB] `Function HasLineInfo() As Boolean Implements`

`IXmlLineInfo.HasLineInfo`

[JScript] `function IXmlLineInfo.HasLineInfo() : Boolean;`

`XmlTextWriter` class (`System.Xml`)

`ToString`

### **Description**

Represents a writer that provides a fast, non-cached, forward-only way of generating streams or files containing XML data that conforms to the W3C Extensible Markup Language (XML) 1.0 and the Namespaces in XML recommendations.

This class implements the `System.Xml.XmlWriter` class.

`XmlTextWriter`

#### **Example Syntax:**

`ToString`

**[C#]** `public XmlTextWriter(TextWriter w);`

**[C++]** `public: XmlTextWriter(TextWriter* w);`

**[VB]** `Public Sub New(ByVal w As TextWriter)`

**[JScript]** `public function XmlTextWriter(w : TextWriter);`

### **Description**

Creates an instance of the `XmlTextWriter` class using the specified `System.IO.TextWriter`. The `TextWriter` to write to. It is assumed that the `TextWriter` is already set to the correct encoding.

`XmlTextWriter`

#### **Example Syntax:**

`ToString`

1  
2 **[C#] public XmlTextWriter(Stream w, Encoding encoding);**

3 **[C++] public: XmlTextWriter(Stream\* w, Encoding\* encoding);**

4 **[VB] Public Sub New(ByVal w As Stream, ByVal encoding As Encoding)**

5 **[JScript] public function XmlTextWriter(w : Stream, encoding : Encoding);**

6 **Creates an instance of the XmlTextWriter class.**

7  
8 ***Description***

9 **Creates an instance of the XmlTextWriter class using the specified**  
10 **stream. The stream to which you want to write. The encoding to generate and**  
11 **whether or not to write out any byte order mark. If encoding is null it writes**  
12 **out the stream as UTF-8 and omits the encoding attribute from the**

13 **ProcessingInstruction.**

14 **XmlTextWriter**

15 ***Example Syntax:***

16 **ToString**

17  
18 **[C#] public XmlTextWriter(string filename, Encoding encoding);**

19 **[C++] public: XmlTextWriter(String\* filename, Encoding\* encoding);**

20 **[VB] Public Sub New(ByVal filename As String, ByVal encoding As**  
21 **Encoding)**

22 **[JScript] public function XmlTextWriter(filename : String, encoding :**  
23 **Encoding);**

24  
25 ***Description***

Creates an instance of the `XmlTextWriter` class using the specified file.  
The filename to write to. If the file exists, it will truncate it and overwrite it with the new content. The encoding to generate. If encoding is null it writes the file out as UTF-8, and omits the encoding attribute from the `ProcessingInstruction`.

Formatting

ToString

[C#] public Formatting Formatting {get; set;}

[C++] public: \_\_property Formatting get\_Formatting();public: \_\_property void set\_Formatting(Formatting);

[VB] Public Property Formatting As Formatting

[JScript] public function get Formatting() : Formatting;public function set Formatting(Formatting);

### *Description*

Indicates how the output is formatted.

If the Indented option is set, child elements are indented using the `System.Xml.XmlTextWriter.Indentation` and `System.Xml.XmlTextWriter.IndentChar` properties. Only element content will be indented. The following C# code writes out HTML elements including mixed content: `XmlTextWriter w = new XmlTextWriter(Console.Out);`  
`w.Formatting = Formatting.Indented; w.WriteStartElement("ol");`  
`w.WriteStartElement("li"); w.WriteString("The big "); // This means "li"`  
`now has a mixed content model. w.WriteElementString("b", "E");`



1 w.WriteElementString("i", "lephant"); w.WriteString(" walks slowly.");  
2 w.WriteEndElement(); w.WriteEndElement(); The above code produces the  
3 following output:

4       The big *Elephant* walks slowly.

5       When this is viewed in HTML no whitespace appears between the bold  
6 and italic elements. In fact, in this example, if indenting was added between  
7 these elements the word "Elephant" would be incorrectly broken.

8       Indentation

9       ToString

10  
11 [C#] public int Indentation {get; set;}

12 [C++] public: \_\_property int get\_Indentation();public: \_\_property void  
13 set\_Indentation(int);

14 [VB] Public Property Indentation As Integer

15 [JScript] public function get Indentation() : int;public function set  
16 Indentation(int);

### 17 18 *Description*

19       Gets or sets how many IndentChars to write for each level in the  
20 hierarchy when System.Xml.XmlTextWriter.Formatting is set to  
21 Formatting.Indented .

22       Indentation is performed on following node types: DocumentType,  
23 Element, Comment, ProcessingInstruction, and CDATASection. All other  
24 node types are not affected. The XmlTextWriter does not indent the internal  
25

DTD subset. However, you could do the following to apply formatting to the internal DTD subset.

IndentChar

ToString

[C#] public char IndentChar {get; set;}

[C++] public: \_\_property \_\_wchar\_t get\_IndentChar();public: \_\_property void set\_IndentChar(\_\_wchar\_t);

[VB] Public Property IndentChar As Char

[JScript] public function get IndentChar() : Char;public function set IndentChar(Char);

### *Description*

Gets or sets which character to use for indenting when System.Xml.XmlTextWriter.Formatting is set to Formatting.Indented.

Namespaces

ToString

[C#] public bool Namespaces {get; set;}

[C++] public: \_\_property bool get\_Namespaces();public: \_\_property void set\_Namespaces(bool);

[VB] Public Property Namespaces As Boolean

[JScript] public function get Namespaces() : Boolean;public function set Namespaces(Boolean);

**Description**

Gets or sets a value indicating whether to do namespace support.

QuoteChar

ToString

**[C#] public char QuoteChar {get; set;}**

**[C++] public: \_\_property \_\_wchar\_t get\_QuoteChar();public: \_\_property**

**void set\_QuoteChar(\_\_wchar\_t);**

**[VB] Public Property QuoteChar As Char**

**[JScript] public function get QuoteChar() : Char;public function set**

**QuoteChar(Char);**

**Description**

Gets or sets which character to use to quote attribute values.

WriteState

ToString

**[C#] public override WriteState WriteState {get;}**

**[C++] public: \_\_property virtual WriteState get\_WriteState();**

**[VB] Overrides Public ReadOnly Property WriteState As WriteState**

**[JScript] public function get WriteState() : WriteState;**

**Description**

Gets the state of the writer.

XmlLang

ToString

[C#] public override string XmlLang {get;}

[C++] public: \_\_property virtual String\* get\_XmlLang();

[VB] Overrides Public ReadOnly Property XmlLang As String

[JScript] public function get XmlLang() : String;

### *Description*

Gets the current xml:lang scope.

This property allows one component to find out what state another component has left the writer in. For example, perhaps one component wants to tell another which language help text to generate. The language information is communicated by writing an xml:lang attribute.

XmlSpace

ToString

[C#] public override XmlSpace XmlSpace {get;}

[C++] public: \_\_property virtual XmlSpace get\_XmlSpace();

[VB] Overrides Public ReadOnly Property XmlSpace As XmlSpace

[JScript] public function get XmlSpace() : XmlSpace;

### *Description*

Gets an System.Xml.XmlSpace representing the current xml:space scope.

1       **This allows one component to find out in what state another**  
2 **component has left the writer.**

3       Close

4  
5 **[C#] public override void Close();**

6 **[C++] public: void Close();**

7 **[VB] Overrides Public Sub Close()**

8 **[JScript] public override function Close();**

9  
10 ***Description***

11       **Closes this stream and the underlying stream.**

12       **Any elements or attributes left open will be automatically closed.**

13       Flush

14  
15 **[C#] public override void Flush();**

16 **[C++] public: void Flush();**

17 **[VB] Overrides Public Sub Flush()**

18 **[JScript] public override function Flush();**

19  
20 ***Description***

21       **Flushes whatever is in the buffer to the underlying streams and also**  
22 **flushes the underlying stream.**

23       **This is called instead of System.Xml.XmlTextWriter.Close when you**  
24 **want to write more to the underlying stream without losing what is still in the**  
25 **buffer.**

## LookupPrefix

[C#] public override string LookupPrefix(string ns);  
[C++] public: String\* LookupPrefix(String\* ns);  
[VB] Overrides Public Function LookupPrefix(ByVal ns As String) As String  
[JScript] public override function LookupPrefix(ns : String) : String;

### *Description*

Returns the closest prefix defined in the current namespace scope for the namespace URI.

**Return Value:** The matching prefix. Or null if no matching namespace URI is found in the current scope. Namespace URI whose prefix you want to find.

## WriteBase64

[C#] public override void WriteBase64(byte[] buffer, int index, int count);  
[C++] public: void WriteBase64(unsigned char buffer \_\_gc[], int index, int count);  
[VB] Overrides Public Sub WriteBase64(ByVal buffer() As Byte, ByVal index As Integer, ByVal count As Integer)  
[JScript] public override function WriteBase64(buffer : Byte[], index : int, count : int);

### *Description*

Encodes the specified binary bytes as base64 and writes out the resulting text.

For example, the byte buffer may contain the binary contents of a GIF image. This clearly would not be valid XML. The Base64 encoding is designed to represent arbitrary byte sequences in a text form comprised of the 65 US-ASCII characters ([A-Za-z0-9+/=]) where each character encodes 6 bits of the binary data. See RFC 1521. (You can obtain RFCs from the Request for Comments Web site at <http://www.rfc-editor.org/>.) The following example encodes an input file using WriteBase64 and generates a temporary XML file. The temporary XML file is decoded using the System.Xml.XmlTextReader.ReadBase64(System.Byte[],System.Int32,System.Int32) method and compared to the original file. Byte array to encode. The position within the buffer indicating the start of the bytes to write. The number of bytes to write.

WriteBinHex

```
[C#] public override void WriteBinHex(byte[] buffer, int index, int count);  
[C++] public: void WriteBinHex(unsigned char buffer __gc[], int index, int  
count);  
[VB] Overrides Public Sub WriteBinHex(ByVal buffer() As Byte, ByVal  
index As Integer, ByVal count As Integer)  
[JScript] public override function WriteBinHex(buffer : Byte[], index : int,  
count : int);
```

### *Description*

1 Encodes the specified binary bytes as binhex and writes out the  
2 resulting text. Byte array to encode. The position within the buffer indicating  
3 the start of the bytes to write. The number of bytes to write.

4 WriteCData

5  
6 [C#] public override void WriteCData(string text);

7 [C++] public: void WriteCData(String\* text);

8 [VB] Overrides Public Sub WriteCData(ByVal text As String)

9 [JScript] public override function WriteCData(text : String);

10  
11 **Description**

12 Writes out a block containing the specified text.

13 If *text* is either null or String.Empty, this method writes an empty

14 CData block, for example . Text to place inside the CDATA block.

15 WriteCharEntity

16  
17 [C#] public override void WriteCharEntity(char ch);

18 [C++] public: void WriteCharEntity(\_\_wchar\_t ch);

19 [VB] Overrides Public Sub WriteCharEntity(ByVal ch As Char)

20 [JScript] public override function WriteCharEntity(ch : Char);

21  
22 **Description**

23 Forces the generation of a character entity for the specified Unicode  
24 character value.



1       **This method writes the Unicode character in hexadecimal character**  
2 **entity reference format. Unicode character for which to generate a character**  
3 **entity.**

4       WriteChars

5  
6 **[C#] public override void WriteChars(char[] buffer, int index, int count);**  
7 **[C++] public: void WriteChars(\_\_wchar\_t buffer \_\_gc[], int index, int count);**  
8 **[VB] Overrides Public Sub WriteChars(ByVal buffer() As Char, ByVal index**  
9 **As Integer, ByVal count As Integer)**  
10 **[JScript] public override function WriteChars(buffer : Char[], index : int,**  
11 **count : int);**

12  
13 ***Description***

14       **Writes text a buffer at a time.**

15       **This method can be used to write large amounts of text a buffer at a**  
16 **time. Character array containing the text to write. The position within the**  
17 **buffer indicating the start of the text to write. The number of characters to**  
18 **write.**

19       WriteComment

20  
21 **[C#] public override void WriteComment(string text);**  
22 **[C++] public: void WriteComment(String\* text);**  
23 **[VB] Overrides Public Sub WriteComment(ByVal text As String)**  
24 **[JScript] public override function WriteComment(text : String);**  
25

1  
2 **Description**

3 Writes out a comment containing the specified text.

4 If *text* is either null or String.Empty, this method writes a Comment  
5 with no data content, for example . Text to place inside the comment.

6 WriteDocType

7  
8 [C#] public override void WriteDocType(string name, string pubid, string  
9 sysid, string subset);

10 [C++] public: void WriteDocType(String\* name, String\* pubid, String\* sysid,  
11 String\* subset);

12 [VB] Overrides Public Sub WriteDocType(ByVal name As String, ByVal  
13 pubid As String, ByVal sysid As String, ByVal subset As String)

14 [JScript] public override function WriteDocType(name : String, pubid :  
15 String, sysid : String, subset : String);

16  
17 **Description**

18 Writes the DOCTYPE declaration with the specified name and  
19 optional attributes.

20 This method does not check for invalid characters in *pubid*, *sysid* or  
21 *subset* . The name of the DOCTYPE. This must be non-empty. If non-null it  
22 also writes PUBLIC "pubid" "sysid" where pubid and sysid are replaced  
23 with the value of the given arguments. If pubid is null and sysid is non-null it  
24 writes SYSTEM "sysid" where sysid is replaced with the value of this  
25

argument. If non-null it writes [subset] where subset is replaced with the value of this argument.

WriteEndAttribute

[C#] public override void WriteEndAttribute();

[C++] public: void WriteEndAttribute();

[VB] Overrides Public Sub WriteEndAttribute()

[JScript] public override function WriteEndAttribute();

### *Description*

Closes the previous System.Xml.XmlTextWriter.WriteStartAttribute(System.String, System.String, System.String) call.

If you call WriteStartAttribute , you can close the attribute with this method.

WriteEndDocument

[C#] public override void WriteEndDocument();

[C++] public: void WriteEndDocument();

[VB] Overrides Public Sub WriteEndDocument()

[JScript] public override function WriteEndDocument();

### *Description*

Closes any open elements or attributes and puts the writer back in the Start state.

## WriteEndElement

**[C#] public override void WriteEndElement();**  
**[C++] public: void WriteEndElement();**  
**[VB] Overrides Public Sub WriteEndElement()**  
**[JScript] public override function WriteEndElement();**

### *Description*

**Closes one element and pops the corresponding namespace scope.**  
**If the element contains no content a short end tag ">" is written;**  
**otherwise a full end tag is written.**

## WriteEntityRef

**[C#] public override void WriteEntityRef(string name);**  
**[C++] public: void WriteEntityRef(String\* name);**  
**[VB] Overrides Public Sub WriteEntityRef(ByVal name As String)**  
**[JScript] public override function WriteEntityRef(name : String);**

### *Description*

**Writes out an entity reference as follows: & name ;. Name of the entity reference.**

## WriteFullEndElement

**[C#] public override void WriteFullEndElement();**  
**[C++] public: void WriteFullEndElement();**

**[VB] Overrides Public Sub WriteFullEndElement()**

**[JScript] public override function WriteFullEndElement();**

### ***Description***

**Closes one element and pops the corresponding namespace scope.**

**This method always writes the full end tag. This is useful when dealing with elements that must include a full end tag. For example, browsers expect HTML script blocks to be closed with "".**

**WriteName**

**[C#] public override void WriteName(string name);**

**[C++] public: void WriteName(String\* name);**

**[VB] Overrides Public Sub WriteName(ByVal name As String)**

**[JScript] public override function WriteName(name : String);**

### ***Description***

**Writes out the specified name, ensuring it is a valid name according to the W3C XML 1.0 recommendation(<http://www.w3.org/TR/1998/REC-xml-19980210#NT-Name> ).**

**If System.Xml.XmlTextWriter.Namespaces is set to true , WriteName also checks that the name is also valid according to the W3C Namespaces in XML recommendation. Name to write.**

**WriteNmToken**

**[C#] public override void WriteNmToken(string name);**

1 [C++] public: void WriteNmToken(String\* name);

2 [VB] Overrides Public Sub WriteNmToken(ByVal name As String)

3 [JScript] public override function WriteNmToken(name : String);

4  
5 **Description**

6 Writes out the specified name, ensuring it is a valid NmToken  
7 according to the W3C XML 1.0  
8 recommendation([http://www.w3.org/TR/1998/REC-xml-19980210#NT-](http://www.w3.org/TR/1998/REC-xml-19980210#NT-Name)  
9 Name). Name to write.

10 WriteProcessingInstruction

11  
12 [C#] public override void WriteProcessingInstruction(string name, string  
13 text);

14 [C++] public: void WriteProcessingInstruction(String\* name, String\* text);

15 [VB] Overrides Public Sub WriteProcessingInstruction(ByVal name As  
16 String, ByVal text As String)

17 [JScript] public override function WriteProcessingInstruction(name : String,  
18 text : String);

19  
20 **Description**

21 Writes out a processing instruction with a space between the name and  
22 text as follows: .

23 If *text* is either null or String.Empty, this method writes a  
24 ProcessingInstruction with no data content, for example . Name of the  
25 processing instruction. Text to include in the processing instruction.

## WriteQualifiedName

```
[C#] public override void WriteQualifiedName(string localName, string ns);
[C++] public: void WriteQualifiedName(String* localName, String* ns);
[VB] Overrides Public Sub WriteQualifiedName(ByVal localName As String,
ByVal ns As String)
[JScript] public override function WriteQualifiedName(localName : String,
ns : String);
```

### *Description*

Writes out the namespace-qualified name. This method looks up the prefix that is in scope for the given namespace.

For example, the following C# code: `writer.Formatting = Formatting.Indented; writer.WriteStartElement("root"); writer.WriteAttributeString("xmlns", "x", null, "urn:abc"); writer.WriteStartElement("item"); writer.WriteStartAttribute("href", null); writer.WriteString("#"); writer.WriteQualifiedName("test", "urn:abc"); writer.WriteEndAttribute(); writer.WriteEndElement(); writer.WriteEndElement(); writer.Close();` Generates the following output: If *ns* maps to the current default namespace, no prefix is generated. The local name to write. The namespace URI to associate with the name.

## WriteRaw

```
[C#] public override void WriteRaw(string data);
[C++] public: void WriteRaw(String* data);
```

1 **[VB] Overrides Public Sub WriteRaw(ByVal data As String)**

2 **[JScript] public override function WriteRaw(data : String);**

3  
4 ***Description***

5 **Writes raw markup manually from a string.**

6 **This method bypasses the entitization of special characters. String**  
7 **containing the text to write.**

8 **WriteRaw**

9  
10 **[C#] public override void WriteRaw(char[] buffer, int index, int count);**

11 **[C++] public: void WriteRaw(\_\_wchar\_t buffer \_\_gc[], int index, int count);**

12 **[VB] Overrides Public Sub WriteRaw(ByVal buffer() As Char, ByVal index**  
13 **As Integer, ByVal count As Integer)**

14 **[JScript] public override function WriteRaw(buffer : Char[], index : int,**  
15 **count : int); Writes raw markup manually.**

16  
17 ***Description***

18 **Writes raw markup manually from a character buffer.**

19 **This method bypasses the entitization of special characters. Character**  
20 **array containing the text to write. The position within the buffer indicating**  
21 **the start of the text to write. The number of characters to write.**

22 **WriteStartAttribute**

23  
24 **[C#] public override void WriteStartAttribute(string prefix, string**  
25 **localName, string ns);**



```

1 [C++] public: void WriteStartAttribute(String* prefix, String* localName,
2 String* ns);
3 [VB] Overrides Public Sub WriteStartAttribute(ByVal prefix As String,
4 ByVal localName As String, ByVal ns As String)
5 [JScript] public override function WriteStartAttribute(prefix : String,
6 localName : String, ns : String); Writes the start of an attribute.

```

### **Description**

**Writes the start of an attribute.**

**This is a more advanced version of**  
**System.Xml.XmlWriter.WriteAttributeString(System.String, System.String, S**  
**ystem.String) that allows you to write an attribute value using multiple write**  
**methods, such as System.Xml.XmlTextWriter.WriteString(System.String) .**  
**Namespace prefix of the attribute. LocalName of the attribute.**  
**NamespaceURI of the attribute**

**WriteStartDocument**

```

18 [C#] public override void WriteStartDocument();
19 [C++] public: void WriteStartDocument();
20 [VB] Overrides Public Sub WriteStartDocument()
21 [JScript] public override function WriteStartDocument(); Writes the XML
22 declaration with the version "1.0".

```

### **Description**

**Writes the XML declaration with the version "1.0".**

1       The encoding level of the document is determined by how the writer is  
2 implemented. For example, if an System.Text.Encoding object is specified in  
3 the XmlTextWriter constructor, this determines the value of the encoding  
4 attribute. This method does not create a standalone attribute.

5       WriteStartDocument

6  
7 [C#] public override void WriteStartDocument(bool standalone);

8 [C++] public: void WriteStartDocument(bool standalone);

9 [VB] Overrides Public Sub WriteStartDocument(ByVal standalone As  
10 Boolean)

11 [JScript] public override function WriteStartDocument(standalone :  
12 Boolean);

13  
14 *Description*

15       Writes the XML declaration with the version "1.0" and the standalone  
16 attribute.

17       The encoding level of the document is determined by how the writer is  
18 implemented. For example, if an System.Text.Encoding object is specified in  
19 the XmlTextWriter constructor, this determines the value of the encoding  
20 attribute. If true, it writes "standalone=yes"; if false, it writes  
21 "standalone=no"

22       WriteStartElement

23  
24 [C#] public override void WriteStartElement(string prefix, string localName,  
25 string ns);

```

1 [C++] public: void WriteStartElement(String* prefix, String* localName,
2 String* ns);
3 [VB] Overrides Public Sub WriteStartElement(ByVal prefix As String, ByVal
4 localName As String, ByVal ns As String)
5 [JScript] public override function WriteStartElement(prefix : String,
6 localName : String, ns : String); Writes the specified start tag.

```

### ***Description***

Writes the specified start tag and associates it with the given namespace and prefix.

After calling this method you can either write attributes or create content using `System.Xml.XmlTextWriter.WriteComment(System.String)`, `System.Xml.XmlTextWriter.WriteString(System.String)`, or `System.Xml.XmlTextWriter.WriteStartElement(System.String, System.String, System.String)` for child elements. You must close this element with `System.Xml.XmlTextWriter.WriteEndElement` at which time an end tag is written out. The namespace prefix of the element. The local name of the element. The namespace URI to associate with the element. If this namespace is already in scope and has an associated prefix then the writer will automatically write that prefix also.

WriteString

```

23 [C#] public override void WriteString(string text);
24 [C++] public: void WriteString(String* text);
25 [VB] Overrides Public Sub WriteString(ByVal text As String)

```

1 **[JScript] public override function WriteString(text : String);**

3 ***Description***

4 **Writes the given text content.**

5 **WriteString does the following The characters & , < , and > are**  
6 **replaced with & , < , and > , respectively. Text to write.**

7 **WriteSurrogateCharEntity**

9 **[C#] public override void WriteSurrogateCharEntity(char lowChar, char**  
10 **highChar);**

11 **[C++] public: void WriteSurrogateCharEntity(\_\_wchar\_t lowChar,**  
12 **\_\_wchar\_t highChar);**

13 **[VB] Overrides Public Sub WriteSurrogateCharEntity(ByVal lowChar As**  
14 **Char, ByVal highChar As Char)**

15 **[JScript] public override function WriteSurrogateCharEntity(lowChar :**  
16 **Char, highChar : Char);**

18 ***Description***

19 **Generates and writes the surrogate character entity for the surrogate**  
20 **character pair.**

21 **The surrogate character entity is written in hexadecimal format. The**  
22 **range for surrogate characters is #x10000 to #x10FFFF. The following**  
23 **formula is used to generate the surrogate character entity: ( *highChar* -**  
24 **0xD800) \* 0x400 + ( *lowChar* -0xDC00) + 0x10000 Applications encode DOM**  
25 **strings using UTF-16. For both HTML and XML, the document character set**

(and therefore the notation of numeric character references) is based on UCS [ISO-10646]. A single numeric character reference in a source document may therefore in some cases correspond to two 16-bit units in a DOM string (a high surrogate and a low surrogate). These 16-bit units are referred to as a surrogate pair. The low surrogate. This must be a value between 0xDC00 and 0xDFFF. The high surrogate. This must be a value between 0xD800 and 0xDBFF.

WriteWhitespace

[C#] public override void WriteWhitespace(string ws);

[C++] public: void WriteWhitespace(String\* ws);

[VB] Overrides Public Sub WriteWhitespace(ByVal ws As String)

[JScript] public override function WriteWhitespace(ws : String);

### *Description*

Writes out the given whitespace.

This method is used to manually format your document. Use the `System.Xml.XmlTextWriter.Formatting` property to have the writer format the output automatically. The string of whitespace characters.

XmlTokenizedType enumeration (System.Xml)

WriteWhitespace

### *Description*

1 Represents the XML type for the string. This allows the string to be  
2 read as a particular XML type, for example a CDATA section type.

3 The XML types are based on the W3C Extensible Markup Language  
4 (XML) 1.0 recommendation.

5 WriteWhitespace

6  
7 [C#] public const XmlTokenizedType CDATA;

8 [C++] public: const XmlTokenizedType CDATA;

9 [VB] Public Const CDATA As XmlTokenizedType

10 [JScript] public var CDATA : XmlTokenizedType;

11  
12 *Description*

13 CDATA type.

14 WriteWhitespace

15  
16 [C#] public const XmlTokenizedType ENTITIES;

17 [C++] public: const XmlTokenizedType ENTITIES;

18 [VB] Public Const ENTITIES As XmlTokenizedType

19 [JScript] public var ENTITIES : XmlTokenizedType;

20  
21 *Description*

22 ENTITIES type.

23 WriteWhitespace

24  
25 [C#] public const XmlTokenizedType ENTITY;

**[C++] public: const XmlTokenizedType ENTITY;**

**[VB] Public Const ENTITY As XmlTokenizedType**

**[JScript] public var ENTITY : XmlTokenizedType;**

***Description***

**ENTITY type.**

WriteWhitespace

**[C#] public const XmlTokenizedType ENUMERATION;**

**[C++] public: const XmlTokenizedType ENUMERATION;**

**[VB] Public Const ENUMERATION As XmlTokenizedType**

**[JScript] public var ENUMERATION : XmlTokenizedType;**

***Description***

**ENUMERATION type.**

WriteWhitespace

**[C#] public const XmlTokenizedType ID;**

**[C++] public: const XmlTokenizedType ID;**

**[VB] Public Const ID As XmlTokenizedType**

**[JScript] public var ID : XmlTokenizedType;**

***Description***

**ID type.**

WriteWhitespace

[C#] public const XmlTokenizedType IDREF;  
 [C++] public: const XmlTokenizedType IDREF;  
 [VB] Public Const IDREF As XmlTokenizedType  
 [JScript] public var IDREF : XmlTokenizedType;

**Description**

IDREF type.

WriteWhitespace

[C#] public const XmlTokenizedType IDREFS;  
 [C++] public: const XmlTokenizedType IDREFS;  
 [VB] Public Const IDREFS As XmlTokenizedType  
 [JScript] public var IDREFS : XmlTokenizedType;

**Description**

IDREFS type.

WriteWhitespace

[C#] public const XmlTokenizedType NCName;  
 [C++] public: const XmlTokenizedType NCName;  
 [VB] Public Const NCName As XmlTokenizedType  
 [JScript] public var NCName : XmlTokenizedType;

**Description**



**NCName type.**

**WriteWhitespace**

**[C#] public const XmlTokenizedType NMTOKEN;**

**[C++] public: const XmlTokenizedType NMTOKEN;**

**[VB] Public Const NMTOKEN As XmlTokenizedType**

**[JScript] public var NMTOKEN : XmlTokenizedType;**

***Description***

**NMTOKEN type.**

**WriteWhitespace**

**[C#] public const XmlTokenizedType NMTOKENS;**

**[C++] public: const XmlTokenizedType NMTOKENS;**

**[VB] Public Const NMTOKENS As XmlTokenizedType**

**[JScript] public var NMTOKENS : XmlTokenizedType;**

***Description***

**NMTOKENS type.**

**WriteWhitespace**

**[C#] public const XmlTokenizedType None;**

**[C++] public: const XmlTokenizedType None;**

**[VB] Public Const None As XmlTokenizedType**

**[JScript] public var None : XmlTokenizedType;**

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25

**Description**

**No type.**  
WriteWhitespace

```
[C#] public const XmlTokenizedType NOTATION;
[C++] public: const XmlTokenizedType NOTATION;
[VB] Public Const NOTATION As XmlTokenizedType
[JScript] public var NOTATION : XmlTokenizedType;
```

**Description**

**NOTATION type.**  
WriteWhitespace

```
[C#] public const XmlTokenizedType QName;
[C++] public: const XmlTokenizedType QName;
[VB] Public Const QName As XmlTokenizedType
[JScript] public var QName : XmlTokenizedType;
```

**Description**

**QName type.**  
XmlUrlResolver class (System.Xml)  
ToString

### ***Description***

**Resolves external XML resources named by a URI.**

**XmlUriResolver is used to resolve external XML resources such as entities, DTDs or schemas. It is also used to process include and import elements found in Extensible StyleSheet Language (XSL) stylesheets or XML Schema Definition language (XSD) schemas.**

**XmlUriResolver**

### ***Example Syntax:***

**ToString**

**[C#] public XmlUriResolver();**

**[C++] public: XmlUriResolver();**

**[VB] Public Sub New()**

**[JScript] public function XmlUriResolver(); Creates a new instance of the XmlUriResolver class.**

### ***Description***

**Creates a new instance of the XmlUriResolver class.**

**Credentials**

**ToString**

**[C#] ICredentials Credentials {set;}**

**[C++] public: \_\_property virtual void set\_Credentials(ICredentials\*);**

1 **[VB] Property Credentials As ICredentials**

2 **[JScript] public function set Credentials(ICredentials);**

3  
4 ***Description***

5 **Sets credentials used to authenticate Web requests.**

6 **If the virtual directory is configured to allow anonymous access, this**  
7 **property does not need to be set. Otherwise, the credentials of the user must**  
8 **be supplied. If credentials are needed but not supplied, XmlUrlResolver uses**  
9 **default credentials (CredentialCache.DefaultCredential).**

10 **GetEntity**

11  
12 **[C#] public override object GetEntity(Uri absoluteUri, string role, Type**  
13 **ofObjectToReturn);**

14 **[C++] public: Object\* GetEntity(Uri\* absoluteUri, String\* role, Type\***  
15 **ofObjectToReturn);**

16 **[VB] Overrides Public Function GetEntity(ByVal absoluteUri As Uri, ByVal**  
17 **role As String, ByVal ofObjectToReturn As Type) As Object**

18 **[JScript] public override function GetEntity(absoluteUri : Uri, role : String,**  
19 **ofObjectToReturn : Type) : Object;**

20  
21 ***Description***

22 **Maps a URI to an object containing the actual resource.**

23 ***Return Value:* A System.IO.Stream object or null if a type other than stream**  
24 **is specified.**

This method is used when the caller wants to map a given URI into an object containing the actual resource that the URI represents. The URI returned from `System.Xml.XmlUrlResolver.ResolveUri(System.Uri, System.String)` The current implementation does not use this parameter when resolving URIs. This is provided for future extensibility purposes. For example, this can be mapped to the `xlink:role` and used as an implementation specific argument in other scenarios. The type of object to return. The current implementation only returns `System.IO.Stream` objects.

#### ResolveUri

**[C#]** public override Uri ResolveUri(Uri baseUri, string relativeUri);

**[C++]** public: Uri\* ResolveUri(Uri\* baseUri, String\* relativeUri);

**[VB]** Overrides Public Function ResolveUri(ByVal baseUri As Uri, ByVal relativeUri As String) As Uri

**[JScript]** public override function ResolveUri(baseUri : Uri, relativeUri : String) : Uri;

#### *Description*

Resolves the absolute URI from the base and relative URIs.

**Return Value:** The absolute URI or null if the relative URI can not be resolved.

The absolute URI may be used as the base URI for any subsequent requests for entities that are relative to this URI. The base URI used to resolve the relative URI The URI to resolve. The URI can be absolute or

1 **relative. If absolute, this value effectively replaces the *baseUri* value. If**  
2 **relative, it combines with the *baseUri* to make an absolute URI.**

3 XmlValidatingReader class (System.Xml)

4 ToString

### 7 ***Description***

8 **Represents a reader that provides DTD, XML-Data Reduced**  
9 **Language (XDR), and XML Schema Definition language (XSD) schema**  
10 **validation.**

11 **XmlValidatingReader implements the System.Xml.XmlReader class**  
12 **and provides support for data validation. Use the**  
13 **System.Xml.XmlValidatingReader.Schemas property to have the reader**  
14 **validate using schema files cached in an**  
15 **System.Xml.Schema.XmlSchemaCollection . The**  
16 **System.Xml.XmlValidatingReader.ValidationType property specifies what**  
17 **type of validation the reader should perform. Setting the property to**  
18 **ValidationType.None creates a non-validating reader.**

19 XmlValidatingReader

### 20 ***Example Syntax:***

21 ToString

22  
23 **[C#] public XmlValidatingReader(XmlReader reader);**

24 **[C++] public: XmlValidatingReader(XmlReader\* reader);**

25 **[VB] Public Sub New(ByVal reader As XmlReader)**

1 **[JScript] public function XmlValidatingReader(reader : XmlReader);**

2 **Initializes a new instance of the XmlValidatingReader class.**

3  
4 ***Description***

5 **Initializes a new instance of the XmlValidatingReader class that**  
6 **validates the content returned from the given System.Xml.XmlReader .**

7 **All nodes returned from the given XmlReader are also returned from**  
8 **this validating reader, so there is no information loss in the process. New**  
9 **nodes not returned from the underlying reader may be added by this reader**  
10 **(for example, default attributes and the children of an entity reference). Any**  
11 **properties set on the given XmlTextReader also applies to this validating**  
12 **reader. For example, if the supplied reader had WhitespaceHandling.None**  
13 **set, this validating reader also ignores whitespace. The XmlReader to read**  
14 **from while validating. The current implementation only supports**  
15 **System.Xml.XmlTextReader .**

16 **XmlValidatingReader**

17 ***Example Syntax:***

18 **ToString**

19  
20 **[C#] public XmlValidatingReader(Stream xmlFragment, XmlNodeType**  
21 **fragType, XmlParserContext context);**

22 **[C++] public: XmlValidatingReader(Stream\* xmlFragment, XmlNodeType**  
23 **fragType, XmlParserContext\* context);**

24 **[VB] Public Sub New(ByVal xmlFragment As Stream, ByVal fragType As**  
25 **XmlNodeType, ByVal context As XmlParserContext)**

```
1 [JScript] public function XmlValidatingReader(xmlFragment : Stream,
2 fragType : XmlNodeType, context : XmlParserContext);
```

#### 4 *Description*

5       **Initializes a new instance of the XmlValidatingReader class with the**  
6 **specified values.**

7       **This constructor parses the given string as a fragment of XML. If the**  
8 **XML fragment is an element or attribute, you can bypass the root level rules**  
9 **for well-formed XML documents. The stream containing the XML fragment**  
10 **to parse. The System.Xml.XmlNodeType of the XML fragment. This**  
11 **determines what the fragment can contain (see table below). The**  
12 **System.Xml.XmlParserContext in which the XML fragment is to be parsed.**  
13 **This includes the System.Xml.XmlNameTable to use, the namespace scope,**  
14 **the current xml:lang and the xml:space scope.**

15       XmlValidatingReader

#### 16 *Example Syntax:*

17       ToString

```
19 [C#] public XmlValidatingReader(string xmlFragment, XmlNodeType
20 fragType, XmlParserContext context);
```

```
21 [C++] public: XmlValidatingReader(String* xmlFragment, XmlNodeType
22 fragType, XmlParserContext* context);
```

```
23 [VB] Public Sub New(ByVal xmlFragment As String, ByVal fragType As
24 XmlNodeType, ByVal context As XmlParserContext)
```

```
25 [JScript] public function XmlValidatingReader(xmlFragment : String,
```



1 fragType : XmlNodeType, context : XmlParserContext);

2  
3 **Description**

4       **Initializes a new instance of the XmlValidatingReader class with the**  
5 **specified values.**

6       **This constructor parses the given string as a fragment of XML. If the**  
7 **XML fragment is an element or attribute, you can bypass the root level rules**  
8 **for well-formed XML documents. This constructor can handle strings**  
9 **returned from System.Xml.XmlValidatingReader.ReadInnerXml . The string**  
10 **containing the XML fragment to parse. The System.Xml.XmlNodeType of the**  
11 **XML fragment. This also determines what the fragment string can contain**  
12 **(see table below). The System.Xml.XmlParserContext in which the XML**  
13 **fragment is to be parsed. This includes the NameTable to use, the namespace**  
14 **scope, the current xml:lang and the xml:space scope.**

15       AttributeCount

16       ToString

17  
18 **[C#] public override int AttributeCount {get;}**

19 **[C++] public: \_\_property virtual int get\_AttributeCount();**

20 **[VB] Overrides Public ReadOnly Property AttributeCount As Integer**

21 **[JScript] public function get AttributeCount() : int;**

22  
23 **Description**

24       **Gets the number of attributes on the current node.**  
25

This property is relevant to **Element** , **DocumentType** and **XmlDeclaration** nodes only. (Other node types do not have attributes.) The following example reads all the elements on the root node.

BaseURI

ToString

[C#] public override string BaseURI {get;}

[C++] public: \_\_property virtual String\* get\_BaseURI();

[VB] Overrides Public ReadOnly Property BaseURI As String

[JScript] public function get BaseURI() : String;

### ***Description***

**Gets the base URI of the current node.**

A networked XML document is comprised of chunks of data aggregated using various W3C standard inclusion mechanisms and therefore contains nodes that come from different places. DTD entities are an example of this, but this is not limited to DTDs. The base URI tells you where these nodes came from. If there is no base URI for the nodes being returned (for example, they were parsed from an in-memory string), **String.Empty** is returned.

CanResolveEntity

ToString

[C#] public override bool CanResolveEntity {get;}

[C++] public: \_\_property virtual bool get\_CanResolveEntity();

1 **[VB] Overrides Public ReadOnly Property CanResolveEntity As Boolean**

2 **[JScript] public function get CanResolveEntity() : Boolean;**

3  
4 ***Description***

5 **Gets a value indicating whether this reader can parse and resolve**  
6 **entities.**

7 Depth

8 ToString

9  
10 **[C#] public override int Depth {get;}**

11 **[C++] public: \_\_property virtual int get\_Depth();**

12 **[VB] Overrides Public ReadOnly Property Depth As Integer**

13 **[JScript] public function get Depth() : int;**

14  
15 ***Description***

16 **Gets the depth of the current node in the XML document.**

17 Encoding

18 ToString

19  
20 **[C#] public Encoding Encoding {get;}**

21 **[C++] public: \_\_property Encoding\* get\_Encoding();**

22 **[VB] Public ReadOnly Property Encoding As Encoding**

23 **[JScript] public function get Encoding() : Encoding;**

24  
25 ***Description***

Gets the encoding attribute for the document.

When any external reference is read (such as expanding an entity in a DTD file or reading a schema file) the encoding property is set to the encoding value of the external reference. If encoding is not specified in the external reference, and there is no byte-order mark, this defaults to UTF-8.

EntityHandling

ToString

[C#] public EntityHandling EntityHandling {get; set;}

[C++] public: \_\_property EntityHandling get\_EntityHandling();public:

\_\_property void set\_EntityHandling(EntityHandling);

[VB] Public Property EntityHandling As EntityHandling

[JScript] public function get EntityHandling() : EntityHandling;public  
function set EntityHandling(EntityHandling);

### *Description*

Gets or sets a value that specifies how the reader handles entities.

This property can be changed on the fly and takes affect after the next System.Xml.XmlValidatingReader.Read call.

EOF

ToString

[C#] public override bool EOF {get;}

[C++] public: \_\_property virtual bool get\_EOF();

[VB] Overrides Public ReadOnly Property EOF As Boolean

1 [JScript] public function get EOF() : Boolean;

3 **Description**

4 Gets a value indicating whether the reader is positioned at the end of  
5 the stream.

6 HasAttributes

7 HasValue

8 ToString

11 **Description**

12 Gets a value indicating whether the current node can have a  
13 System.Xml.XmlValidatingReader.Value .

14 The following table lists node types that have a value to return.

15 IsDefault

16 ToString

18 [C#] public override bool IsDefault {get;}

19 [C++] public: \_\_property virtual bool get\_IsDefault();

20 [VB] Overrides Public ReadOnly Property IsDefault As Boolean

21 [JScript] public function get IsDefault() : Boolean;

23 **Description**

24 Gets a value indicating whether the current node is an attribute that  
25 was generated from the default value defined in the DTD or schema.

This property applies only to an attribute node.

IsEmptyElement

ToString

[C#] public override bool IsEmptyElement {get;}

[C++] public: \_\_property virtual bool get\_IsEmptyElement();

[VB] Overrides Public ReadOnly Property IsEmptyElement As Boolean

[JScript] public function get IsEmptyElement() : Boolean;

### *Description*

Gets a value indicating whether the current node is an empty element (for example, ).

This property enables you to determine the difference between the following: ( IsEmptyElement is true ).

Item

ToString

[C#] public override string this[int i] {get;}

[C++] public: \_\_property virtual String\* get\_Item(int i);

[VB] Overrides Public Default ReadOnly Property Item(ByVal i As Integer)

As String

[JScript] returnValue = XmlValidatingReaderObject.Item(i); Gets the value of the specified attribute.

### *Description*

**Gets the value of the attribute with the specified index.**

**This property does not move the reader. The index of the attribute.**

**Item**

**ToString**

**[C#] public override string this[string name] {get;}**

**[C++] public: \_\_property virtual String\* get\_Item(String\* name);**

**[VB] Overrides Public Default ReadOnly Property Item(ByVal name As String) As String**

**[JScript] returnValue = XmlValidatingReaderObject.Item(name);**

### ***Description***

**Gets the value of the attribute with the specified name.**

**This property does not move the reader. The qualified name of the attribute.**

**Item**

**ToString**

**[C#] public override string this[string name, string namespaceURI] {get;}**

**[C++] public: \_\_property virtual String\* get\_Item(String\* name, String\* namespaceURI);**

**[VB] Overrides Public Default ReadOnly Property Item(ByVal name As String, ByVal namespaceURI As String) As String**

**[JScript] returnValue = XmlValidatingReaderObject.Item(name, namespaceURI);**

**Description**

Gets the value of the attribute with the specified local name and namespace URI.

This property does not move the reader. The local name of the attribute. The namespace URI of the attribute.

LocalName

ToString

[C#] public override string LocalName {get;}

[C++] public: \_\_property virtual String\* get\_LocalName();

[VB] Overrides Public ReadOnly Property LocalName As String

[JScript] public function get LocalName() : String;

**Description**

Gets the local name of the current node.

Name

ToString

[C#] public override string Name {get;}

[C++] public: \_\_property virtual String\* get\_Name();

[VB] Overrides Public ReadOnly Property Name As String

[JScript] public function get Name() : String;

**Description**



**Gets the qualified name of the current node.**

Namespaces

ToString

**[C#] public bool Namespaces {get; set;}**

**[C++] public: \_\_property bool get\_Namespaces();public: \_\_property void**

**set\_Namespaces(bool);**

**[VB] Public Property Namespaces As Boolean**

**[JScript] public function get Namespaces() : Boolean;public function set**

**Namespaces(Boolean);**

### ***Description***

**Gets or sets a value indicating whether to do namespace support.**

NamespaceURI

ToString

**[C#] public override string NamespaceURI {get;}**

**[C++] public: \_\_property virtual String\* get\_NamespaceURI();**

**[VB] Overrides Public ReadOnly Property NamespaceURI As String**

**[JScript] public function get NamespaceURI() : String;**

### ***Description***

**Gets the namespace URI (as defined in the W3C Namespace specification) of the node on which the reader is positioned.**

**This property is relevant to Element and Attribute nodes only.**

NameTable

ToString

[C#] public override XmlNameTable NameTable {get;}

[C++] public: \_\_property virtual XmlNameTable\* get\_NameTable();

[VB] Overrides Public ReadOnly Property NameTable As XmlNameTable

[JScript] public function get NameTable() : XmlNameTable;

### **Description**

Gets the System.Xml.XmlNameTable associated with this implementation.

All node and attribute names returned from this class are atomized using the NameTable . When the same name is returned multiple times (for example, Customer ), then the same String object will be returned for that name. This makes it possible for you to write efficient code that does object comparisons on these strings instead of expensive string comparisons.

· NodeType

ToString

[C#] public override XmlNodeType NodeType {get;}

[C++] public: \_\_property virtual XmlNodeType get\_NodeType();

[VB] Overrides Public ReadOnly Property NodeType As XmlNodeType

[JScript] public function get NodeType() : XmlNodeType;

### **Description**

**Gets the type of the current node.**

**This property never returns the following XmlNodeType types:**

**Document , DocumentFragment , Entity , or Notation .**

**Prefix**

**ToString**

**[C#] public override string Prefix {get;}**

**[C++] public: \_\_property virtual String\* get\_Prefix();**

**[VB] Overrides Public ReadOnly Property Prefix As String**

**[JScript] public function get Prefix() : String;**

### ***Description***

**Gets the namespace prefix associated with the current node.**

**QuoteChar**

**ToString**

**[C#] public override char QuoteChar {get;}**

**[C++] public: \_\_property virtual \_\_wchar\_t get\_QuoteChar();**

**[VB] Overrides Public ReadOnly Property QuoteChar As Char**

**[JScript] public function get QuoteChar() : Char;**

### ***Description***

**Gets the quotation mark character used to enclose the value of an attribute node.**

**This property applies only to an attribute node.**

Reader

ToString

[C#] public XmlReader Reader {get;}

[C++] public: \_\_property XmlReader\* get\_Reader();

[VB] Public ReadOnly Property Reader As XmlReader

[JScript] public function get Reader() : XmlReader;

### *Description*

Gets the System.Xml.XmlReader used to construct this XmlValidatingReader .

This property allows the user to access properties specific to the supplied reader. For example, the user could get the underlying the System.Xml.XmlTextReader.LineNumber and System.Xml.XmlTextReader.LinePosition of the underlying reader. This can be useful when reporting validation errors.

ReadState

ToString

[C#] public override ReadState ReadState {get;}

[C++] public: \_\_property virtual ReadState get\_ReadState();

[VB] Overrides Public ReadOnly Property ReadState As ReadState

[JScript] public function get ReadState() : ReadState;

### *Description*

Gets the state of the reader.

Schemas

ToString

[C#] public XmlSchemaCollection Schemas {get;}

[C++] public: \_\_property XmlSchemaCollection\* get\_Schemas();

[VB] Public ReadOnly Property Schemas As XmlSchemaCollection

[JScript] public function get Schemas() : XmlSchemaCollection;

### **Description**

Gets a System.Xml.Schema.XmlSchemaCollection to use for validation.

The XmlSchemaCollection holds pre-loaded XDR and XSD schemas.

This property gives the reader access to the cache of schemas and allows it to validate without having to re-load schemas every time. The reader does not add anything to the XmlSchemaCollection .

SchemaType

ToString

[C#] public object SchemaType {get;}

[C++] public: \_\_property Object\* get\_SchemaType();

[VB] Public ReadOnly Property SchemaType As Object

[JScript] public function get SchemaType() : Object;

### **Description**

Gets a schema type object.

The user needs to test for the returned type. For example object  
obj=vreader.SchemaType; if (obj is XmlSchemaType) XmlSchemaType  
st=obj.BaseSchemaType; if (obj is XmlSchemaDataType) Type  
vt=obj.ValueType; If XSD schema validation is being performed then the  
XmlSchemaType or XmlSchemaDataType corresponds to the current  
element being read. If DTD validation is being performed this property  
returns null .

ValidationType

ToString

[C#] public ValidationType ValidationType {get; set;}

[C++] public: \_\_property ValidationType get\_ValidationType();public:

\_\_property void set\_ValidationType(ValidationType);

[VB] Public Property ValidationType As ValidationType

[JScript] public function get ValidationType() : ValidationType;public

function set ValidationType(ValidationType);

### *Description*

Gets a value describing what type of validation to perform.

This property must be set before the first call to  
System.Xml.XmlValidatingReader.Read . Setting this property to  
ValidationType.None creates a non-validating reader.

Value

ToString

**[C#] public override string Value {get;}**

**[C++] public: \_\_property virtual String\* get\_Value();**

**[VB] Overrides Public ReadOnly Property Value As String**

**[JScript] public function get Value() : String;**

### ***Description***

**Gets the text value of the current node.**

XmlLang

ToString

**[C#] public override string XmlLang {get;}**

**[C++] public: \_\_property virtual String\* get\_XmlLang();**

**[VB] Overrides Public ReadOnly Property XmlLang As String**

**[JScript] public function get XmlLang() : String;**

### ***Description***

**Gets the current xml:lang scope.**

**This property represents the xml:lang scope within which the current node resides. For example, here is an XML fragment with xml:lang set to US English in the root element: Fred When the reader is positioned on the name element, you can use this property to find that it is in the scope of a US English xml:lang attribute.**

XmlResolver

ToString

1  
2 **[C#] XmlResolver XmlResolver {set;}**

3 **[C++] public: \_\_property void set \_XmlResolver(XmlResolver\*);**

4 **[VB] Property XmlResolver As XmlResolver**

5 **[JScript] public function set XmlResolver(XmlResolver);**

6  
7 ***Description***

8       **Sets the System.Xml.XmlResolver used for resolving external DTD and**  
9 **schema location references. The XmlResolver is also used to handle any**  
10 **import or include elements found in XSD schemas.**

11       **The XmlResolver is used to load any DTDs, entities or schemas needed**  
12 **to complete the validation process.**

13       XmlSpace

14       ToString

15  
16 **[C#] public override XmlSpace XmlSpace {get;}**

17 **[C++] public: \_\_property virtual XmlSpace get \_XmlSpace();**

18 **[VB] Overrides Public ReadOnly Property XmlSpace As XmlSpace**

19 **[JScript] public function get XmlSpace() : XmlSpace;**

20  
21 ***Description***

22       **Gets the current xml:space scope.**

23       ToString

24  
25 **[C#] public event ValidationEventHandler ValidationEventHandler;**



1 [C++] public: \_\_event ValidationEventHandler\* ValidationEventHandler;

2 [VB] Public Event ValidationEventHandler As ValidationEventHandler

3  
4 *Description*

5 Sets an event handler for receiving information about DTD, XDR and  
6 XSD schema validation errors.

7 These events occur during System.Xml.XmlValidatingReader.Read  
8 and only if a System.Xml.XmlValidatingReader.ValidationType of DTD,  
9 XDR, Schema, or Auto is specified.

10 Close

11  
12 [C#] public override void Close();

13 [C++] public: void Close();

14 [VB] Overrides Public Sub Close()

15 [JScript] public override function Close();

16  
17 *Description*

18 Changes the System.Xml.XmlReader.ReadState to Closed.

19 This method also releases any resources held while reading. If this  
20 reader was constructed using an System.Xml.XmlTextReader , this method  
21 also calls Close on the underlying reader. If this reader was constructed using  
22 a stream, this method also calls Close on the underlying stream.

23 GetAttribute

24  
25 [C#] public override string GetAttribute(int i);

**[C++] public: String\* GetAttribute(int i);**  
**[VB] Overrides Public Function GetAttribute(ByVal i As Integer) As String**  
**[JScript] public override function GetAttribute(i : int) : String; Gets the**  
**value of an attribute.**

### ***Description***

**Gets the value of the attribute with the specified index.**

***Return Value:*** The value of the specified attribute.

**This method does not move the reader. The index of the attribute. The index is zero-based. (The first attribute has index 0.)**

GetAttribute

**[C#] public override string GetAttribute(string name);**

**[C++] public: String\* GetAttribute(String\* name);**

**[VB] Overrides Public Function GetAttribute(ByVal name As String) As String**

**[JScript] public override function GetAttribute(name : String) : String;**

### ***Description***

**Gets the value of the attribute with the specified name.**

***Return Value:*** The value of the specified attribute. If the attribute is not found, String.Empty is returned.

**This method does not move the reader. The qualified name of the attribute.**

GetAttribute

```

1
2 [C#] public override string GetAttribute(string localName, string
3 namespaceURI);
4 [C++] public: String* GetAttribute(String* localName, String*
5 namespaceURI);
6 [VB] Overrides Public Function GetAttribute(ByVal localName As String,
7 ByVal namespaceURI As String) As String
8 [JScript] public override function GetAttribute(localName : String,
9 namespaceURI : String) : String;
10

```

### *Description*

Gets the value of the attribute with the specified local name and namespace URI.

**Return Value:** The value of the specified attribute. If the attribute is not found, String.Empty is returned. This method does not move the reader.

The following XML contains an attribute in a specific namespace: You can lookup the dt:type attribute using one argument (prefix and local name) or two arguments (local name and namespace URI):

```

String dt =
reader.GetAttribute("dt:type"); String dt2 =
reader.GetAttribute("type","urn:datatypes"); To lookup the xmlns:dt
attribute, use one of the following arguments: String dt3 =
reader.GetAttribute("xmlns:dt"); String dt4 =
reader.GetAttribute("dt",http://www.w3.org/2000/xmlns/); You can also get
this information using the System.Xml.XmlValidatingReader.Prefix

```

property. The local name of the attribute. The namespace URI of the attribute.

LookupNamespace

[C#] public override string LookupNamespace(string prefix);

[C++] public: String\* LookupNamespace(String\* prefix);

[VB] Overrides Public Function LookupNamespace(ByVal prefix As String)  
As String

[JScript] public override function LookupNamespace(prefix : String) :  
String;

### *Description*

Resolves a namespace prefix in the current element's scope.

**Return Value:** The namespace URI to which the prefix maps or null if no matching prefix is found.

In the preceding XML, if the reader is positioned on the href attribute, the prefix a is resolved by calling reader.LookupNamesapce("a") . The returned string is urn:456 . The prefix whose namespace URI you want to resolve. To match the default namespace, pass an empty string. This string does not have to be atomized.

MoveToAttribute

[C#] public override void MoveToAttribute(int i);

[C++] public: void MoveToAttribute(int i);

[VB] Overrides Public Sub MoveToAttribute(ByVal i As Integer)

### Description

**Moves to the attribute with the specified index. The index of the attribute.**

## MoveToAttribute

```
[C#] public override bool MoveToAttribute(string name);
```

```
[C++] public: bool MoveToAttribute(String* name);
```

**[VB] Overrides Public Function MoveToAttribute(ByVal name As String) As Boolean**

**[JScript] public override function MoveToAttribute(name : String) :**

**Boolean; Moves to the specified attribute.**

### Description

**Moves to the attribute with the specified name.**

***Return Value:*** true if the attribute is found; otherwise, false . If false , the position of the reader does not change.

After calling this method, the `System.Xml.XmlValidatingReader.Name`, `System.Xml.XmlValidatingReader.NamespaceURI`, and `System.Xml.XmlValidatingReader.Prefix` properties reflect the properties of that attribute. The qualified name of the attribute.

## MoveToAttribute

```
[C#] public override bool MoveToAttribute(string localName, string
```

```

1 namespaceURI);
2 [C++] public: bool MoveToAttribute(String* localName, String*
3 namespaceURI);
4 [VB] Overrides Public Function MoveToAttribute(ByVal localName As
5 String, ByVal namespaceURI As String) As Boolean
6 [JScript] public override function MoveToAttribute(localName : String,
7 namespaceURI : String) : Boolean;
8

```

### *Description*

Moves to the attribute with the specified local name and namespace URI.

**Return Value:** true if the attribute is found; otherwise, false . If false , the position of the reader does not change.

After calling this method, the System.Xml.XmlValidatingReader.Name , System.Xml.XmlValidatingReader.NamespaceURI , and System.Xml.XmlValidatingReader.Prefix properties reflect the properties of that attribute. The local name of the attribute. The namespace URI of the attribute.

MoveToElement

```

21 [C#] public override bool MoveToElement();
22 [C++] public: bool MoveToElement();
23 [VB] Overrides Public Function MoveToElement() As Boolean
24 [JScript] public override function MoveToElement() : Boolean;
25

```

1  
2 **Description**

3 Moves to the element that contains the current attribute node.

4 **Return Value:** true if the reader is positioned on an attribute (the reader  
5 moves to the element that owns the attribute); false if the reader is not  
6 positioned on an attribute (the position of the reader does not change).

7 Use this method to return to an element after navigating through its  
8 attributes. This method moves the reader to one of the following node types:

9 **Element , DocumentType , or XmlDeclaration .**

10 MoveToFirstAttribute

11  
12 **[C#] public override bool MoveToFirstAttribute();**

13 **[C++] public: bool MoveToFirstAttribute();**

14 **[VB] Overrides Public Function MoveToFirstAttribute() As Boolean**

15 **[JScript] public override function MoveToFirstAttribute() : Boolean;**

16  
17 **Description**

18 Moves to the first attribute.

19 **Return Value:** true if an attribute exists (the reader moves to the first  
20 attribute); otherwise, false (the position of the reader does not change).

21 MoveToNextAttribute

22  
23 **[C#] public override bool MoveToNextAttribute();**

24 **[C++] public: bool MoveToNextAttribute();**

25 **[VB] Overrides Public Function MoveToNextAttribute() As Boolean**

**[JScript] public override function MoveToNextAttribute() : Boolean;**

***Description***

Moves to the next attribute.

***Return Value:*** true if there is a next attribute; false if there are no more attributes.

If the current node is an element node, this method is equivalent to **System.Xml.XmlValidatingReader.MoveToFirstAttribute** . If **MoveToNextAttribute** returns true , the reader moves to the next attribute; otherwise, the position of the reader does not change.

Read

**[C#] public override bool Read();**

**[C++] public: bool Read();**

**[VB] Overrides Public Function Read() As Boolean**

**[JScript] public override function Read() : Boolean;**

***Description***

Reads the next node from the stream.

***Return Value:*** true if the next node was read successfully; false if there are no more nodes to read.

When a reader is first created and initialized, there is no information available. You must call **Read** to read the first node.

ReadAttributeValue



**[C#] public override bool ReadAttributeValue();**

**[C++] public: bool ReadAttributeValue();**

**[VB] Overrides Public Function ReadAttributeValue() As Boolean**

**[JScript] public override function ReadAttributeValue() : Boolean;**

***Description***

Parses the attribute value into one or more Text or EntityReference nodes.

***Return Value:*** true if there are nodes to return.

Use this method after calling System.Xml.XmlValidatingReader.MoveToAttribute(System.String) to read through the text or entity reference nodes that make up the attribute value. The System.Xml.XmlReader.Depth of the attribute value nodes is one plus the depth of the attribute node. The Depth increments and decrements by one when you step into and out of general entity references.

ReadInnerXml

**[C#] public override string ReadInnerXml();**

**[C++] public: String\* ReadInnerXml();**

**[VB] Overrides Public Function ReadInnerXml() As String**

**[JScript] public override function ReadInnerXml() : String;**

***Description***

Reads all the content, including markup, as a string.

**Return Value:** All the XML content, including markup, in the current node. If the current node has no children, an empty string is returned.

This method returns all the content of the current node including the markup. The current node (start tag) and corresponding end node (end tag) are not returned. For example, if you had the following: this ReadInnerXml returns this This method handles element and attribute nodes in the following way: Node Type Child Content Return Value Position After the Call Element text text After the end tag.

ReadOuterXml

[C#] public override string ReadOuterXml();

[C++] public: String\* ReadOuterXml();

[VB] Overrides Public Function ReadOuterXml() As String

[JScript] public override function ReadOuterXml() : String;

### **Description**

Reads the content, including markup, representing this node and all its children.

**Return Value:** If the reader is positioned on an element or an attribute node, this method returns all the XML content, including markup, of the current node and all its children; otherwise, it returns an empty string.

This method is similar to System.Xml.XmlValidatingReader.ReadInnerXml except it also returns the start and end tags.

## ReadString

[C#] public override string ReadString();  
[C++] public: String\* ReadString();  
[VB] Overrides Public Function ReadString() As String  
[JScript] public override function ReadString() : String;

### *Description*

Reads the contents of an element or text node as a string.

**Return Value:** The contents of the element or text node. This can be an empty string if the reader is positioned on something other than an element or text node, or if there is no more text content to return in the current context.

If positioned on an element, ReadString concatenates all text, significant whitespace, whitespace and CDATA section node types together and returns the concatenated data as the element content. The reader stops when any markup is encountered. This could occur in a mixed content model, or when an element end tag is read.

## ReadTypedValue

[C#] public object ReadTypedValue();  
[C++] public: Object\* ReadTypedValue();  
[VB] Public Function ReadTypedValue() As Object  
[JScript] public function ReadTypedValue() : Object;

### *Description*

Gets the common language runtime type for the specified XSD type

For example, if the type is defined as `xsd:int` the the runtime type `int32` is returned for the object. This can be tested using the `System.Object.GetType` method and cast accordingly. This property always return a subclassed type of object. An object of the type object is never returned.

`ResolveEntity`

[C#] `public override void ResolveEntity();`

[C++] `public: void ResolveEntity();`

[VB] `Overrides Public Sub ResolveEntity()`

[JScript] `public override function ResolveEntity();`

### *Description*

Resolves the entity reference for `EntityReference` nodes.

If the reader is positioned on an `EntityReference` node ( `XmlNodeType.EntityReference` ), if `System.Xml.XmlValidatingReader.Read` is called after calling this method, the entity replacement text is parsed. When the entity replacement text is finished, an `EndEntity` node is returned to close the entity reference scope.

`IXmlLineInfo.HasLineInfo`

[C#] `bool IXmlLineInfo.HasLineInfo();`

[C++] `bool IXmlLineInfo::HasLineInfo();`

[VB] `Function HasLineInfo() As Boolean Implements`

## **IXmlLineInfo.HasLineInfo**

**[JScript] function IXmlLineInfo.HasLineInfo() : Boolean;**

XmlWhitespace class (System.Xml)

ToString

### ***Description***

Represents whitespace in element content. These nodes are created automatically at System.Xml.XmlDocument.Load(System.String) time only if the System.Xml.XmlDocument.PreserveWhitespace flag is true .

XmlWhitespace

### ***Example Syntax:***

ToString

**[C#] protected internal XmlWhitespace(string strData, XmlDocument doc);**

**[C++] internal: XmlWhitespace(String\* strData, XmlDocument\* doc);**

**[VB] Protected Friend Sub New(ByVal strData As String, ByVal doc As XmlDocument)**

**[JScript] package function XmlWhitespace(strData : String, doc : XmlDocument);**

### ***Description***

Attributes

BaseURI

ChildNodes

- 1 Data
- 2 FirstChild
- 3 HasChildNodes
- 4 InnerText
- 5 InnerXml
- 6 IsReadOnly
- 7 Item
- 8 Item
- 9 LastChild
- 10 Length
- 11 LocalName
- 12 ToString

**Description**

**Gets the local name of the node.**

Name

ToString

**[C#] public override string Name {get;}**

**[C++] public: \_\_property virtual String\* get\_Name();**

**[VB] Overrides Public ReadOnly Property Name As String**

**[JScript] public function get Name() : String;**

**Description**

Technical Reference

**Gets the qualified name of the node.**

- NamespaceURI
- NextSibling
- NodeType
- ToString

**Description**

**Gets the type of the node.**

- OuterXml
- OwnerDocument
- ParentNode
- Prefix
- PreviousSibling
- Value
- ToString

**Description**

**Gets or sets the value of the node.**

CloneNode

```
[C#] public override XmlNode CloneNode(bool deep);  
[C++] public: XmlNode* CloneNode(bool deep);  
[VB] Overrides Public Function CloneNode(ByVal deep As Boolean) As
```

## **XmlNode**

**[JScript] public override function CloneNode(deep : Boolean) : XmlNode;**

### ***Description***

**Creates a duplicate of this node.**

***Return Value:*** The duplicate node.

**This method serves as a generic copy constructor for nodes. The duplicate node has no parent (ParentNode returns null). true to recursively clone the subtree under the specified node; false to clone only the node itself (and its attributes if the node is an XmlElement).**

### **WriteContentTo**

**[C#] public override void WriteContentTo(XmlWriter w);**

**[C++] public: void WriteContentTo(XmlWriter\* w);**

**[VB] Overrides Public Sub WriteContentTo(ByVal w As XmlWriter)**

**[JScript] public override function WriteContentTo(w : XmlWriter);**

### ***Description***

**Saves all the children of the node to the specified XmlWriter. The XmlWriter where you want to save the current node.**

### **WriteTo**

**[C#] public override void WriteTo(XmlWriter w);**

**[C++] public: void WriteTo(XmlWriter\* w);**

**[VB] Overrides Public Sub WriteTo(ByVal w As XmlWriter)**



1 [JScript] public override function WriteTo(w : XmlWriter);

2  
3 **Description**

4 Saves the node to the specified XmlWriter. The XmlWriter where you  
5 want to save the node.

6 XmlWriter class (System.Xml)

7 WriteTo

8  
9  
10 **Description**

11 Represents a writer that provides a fast, non-cached, forward-only  
12 means of generating streams or files containing XML data that conforms to  
13 the W3C Extensible Markup Language (XML) 1.0 and the Namespaces in  
14 XML recommendations.

15 XmlWriter is implemented in the System.Xml.XmlTextWriter class.

16 XmlWriter

17 **Example Syntax:**

18 WriteTo

19  
20 [C#] protected XmlWriter();

21 [C++] protected: XmlWriter();

22 [VB] Protected Sub New()

23 [JScript] protected function XmlWriter();

24 WriteState

25 WriteTo

1  
2 **[C#] public abstract WriteState WriteState {get;}**

3 **[C++] public: \_\_property virtual WriteState get\_WriteState() = 0;**

4 **[VB] MustOverride Public ReadOnly Property WriteState As WriteState**

5 **[JScript] public abstract function get WriteState() : WriteState;**

6  
7 ***Description***

8 **When overridden in a derived class, gets the state of the writer.**

9 XmlLang

10 WriteTo

11  
12 **[C#] public abstract string XmlLang {get;}**

13 **[C++] public: \_\_property virtual String\* get\_XmlLang() = 0;**

14 **[VB] MustOverride Public ReadOnly Property XmlLang As String**

15 **[JScript] public abstract function get XmlLang() : String;**

16  
17 ***Description***

18 **When overridden in a derived class, gets the current xml:lang scope.**

19 **This property allows one component to find out in what state another**  
20 **component has left the writer. For example, perhaps one component wants to**  
21 **tell another which language help text to generate. The language information**  
22 **is communicated by writing an xml:lang attribute.**

23 XmlSpace

24 WriteTo

1  
2 **[C#] public abstract XmlSpace XmlSpace {get;}**

3 **[C++] public: \_\_property virtual XmlSpace get\_XmlSpace() = 0;**

4 **[VB] MustOverride Public ReadOnly Property XmlSpace As XmlSpace**

5 **[JScript] public abstract function get XmlSpace() : XmlSpace;**

6  
7 ***Description***

8       **When overridden in a derived class, gets an System.Xml.XmlSpace**  
9 **representing the current xml:space scope.**

10       **This allows one component to find out what state another component**  
11 **has left the writer in.**

12       Close

13  
14 **[C#] public abstract void Close();**

15 **[C++] public: virtual void Close() = 0;**

16 **[VB] MustOverride Public Sub Close()**

17 **[JScript] public abstract function Close();**

18  
19 ***Description***

20       **When overridden in a derived class, closes this stream and the**  
21 **underlying stream.**

22       **Any elements or attributes left open will be automatically closed.**

23       Flush

24  
25 **[C#] public abstract void Flush();**

1 [C++] public: virtual void Flush() = 0;  
2 [VB] MustOverride Public Sub Flush()  
3 [JScript] public abstract function Flush();  
4

#### 5 *Description*

6 When overridden in a derived class, flushes whatever is in the buffer to  
7 the underlying streams and also flushes the underlying stream.

8 This is called instead of System.Xml.XmlWriter.Close when you want  
9 to write more to the underlying stream without losing what is still in the  
10 buffer.

11 LookupPrefix

12  
13 [C#] public abstract string LookupPrefix(string ns);  
14 [C++] public: virtual String\* LookupPrefix(String\* ns) = 0;  
15 [VB] MustOverride Public Function LookupPrefix(ByVal ns As String) As  
16 String  
17 [JScript] public abstract function LookupPrefix(ns : String) : String;  
18

#### 19 *Description*

20 When overridden in a derived class, returns the closest prefix defined  
21 in the current namespace scope for the namespace URI.

22 *Return Value:* The matching prefix or null if no matching namespace URI is  
23 found in the current scope. The namespace URI whose prefix you want to  
24 find.

25 WriteAttributes

```

1
2 [C#] public virtual void WriteAttributes(XmlReader reader, bool defattr);
3 [C++] public: virtual void WriteAttributes(XmlReader* reader, bool defattr);
4 [VB] Overridable Public Sub WriteAttributes(ByVal reader As XmlReader,
5 ByVal defattr As Boolean)
6 [JScript] public function WriteAttributes(reader : XmlReader, defattr :
7 Boolean);
8

```

### *Description*

When overridden in a derived class, writes out all the attributes found at the current position in the System.Xml.XmlReader .

If the reader is positioned on an element node WriteAttributes copies all the contained attributes. If the reader is positioned on an attribute node, this method writes the current attribute, then the rest of the attributes until the element closing tag. If the reader is positioned on an XmlDeclaration node, this method writes all the attributes in the declaration. If the reader is positioned on any other node type this method has no operation. The XmlReader from which to copy the attributes. true to copy the default attributes from the XmlReader ; otherwise, false.

WriteAttributeString

```

21
22 [C#] public void WriteAttributeString(string localName, string value);
23 [C++] public: void WriteAttributeString(String* localName, String* value);
24 [VB] Public Sub WriteAttributeString(ByVal localName As String, ByVal
25 value As String)

```

**[JScript] public function WriteAttributeString(localName : String, value : String);**

***Description***

When overridden in a derived class, writes out the attribute with the specified local name and value.

WriteAttributeString does the following If the attribute value includes double or single quotes, they are replaced with " and &apos; respectively.

The local name of the attribute. The value of the attribute.

WriteAttributeString

**[C#] public void WriteAttributeString(string localName, string ns, string value);**

**[C++] public: void WriteAttributeString(String\* localName, String\* ns, String\* value);**

**[VB] Public Sub WriteAttributeString(ByVal localName As String, ByVal ns As String, ByVal value As String)**

**[JScript] public function WriteAttributeString(localName : String, ns : String, value : String);** When overridden in a derived class, writes an attribute with the specified value.

***Description***

When overridden in a derived class, writes an attribute with the specified parameters.

1        This method writes out the attribute with a user defined namespace  
2        prefix and associates it with the given namespace. If *localName* is "xmlns"  
3        then this method also treats this as a namespace declaration. In this case, the  
4        *ns* argument should be null . The local name of the attribute. The namespace  
5        URI to associate with the attribute. The value of the attribute.

6        WriteAttributeString

7  
8        [C#] public void WriteAttributeString(string prefix, string localName, string  
9        ns, string value);

10        [C++] public: void WriteAttributeString(String\* prefix, String\* localName,  
11        String\* ns, String\* value);

12        [VB] Public Sub WriteAttributeString(ByVal prefix As String, ByVal  
13        localName As String, ByVal ns As String, ByVal value As String)

14        [JScript] public function WriteAttributeString(prefix : String, localName :  
15        String, ns : String, value : String);

16  
17        **Description**

18        When overridden in a derived class, writes out the attribute with the  
19        specified prefix, local name, namespace URI and value.

20        This method writes out the attribute with a user defined namespace  
21        prefix and associates it with the given namespace. If the prefix is "xmlns"  
22        then this method also treats this as a namespace declaration and associates  
23        the declared prefix with the namespace URI provided in the given attribute  
24        value. In this case the *ns* argument should be null . The namespace prefix of  
25

1 the attribute. The local name of the attribute. The namespace URI of the  
2 attribute. The value of the attribute.

3 WriteBase64

4  
5 [C#] public abstract void WriteBase64(byte[] buffer, int index, int count);

6 [C++] public: virtual void WriteBase64(unsigned char buffer \_\_gc[], int  
7 index, int count) = 0;

8 [VB] MustOverride Public Sub WriteBase64(ByVal buffer() As Byte, ByVal  
9 index As Integer, ByVal count As Integer)

10 [JScript] public abstract function WriteBase64(buffer : Byte[], index : int,  
11 count : int);

### 12 13 *Description*

14 When overridden in a derived class, encodes the specified binary bytes  
15 as base64 and writes out the resulting text.

16 For example, the byte buffer may contain the binary contents of a GIF  
17 image. This clearly would not be valid XML. The base64 encoding is designed  
18 to represent arbitrary byte sequences in a text form comprised of the 65 US-  
19 ASCII characters ([A-Za-z0-9+/=]) where each character encodes 6 bits of the  
20 binary data. See RFC 1521. (You can obtain RFCs from the Request for  
21 Comments Web site at <http://www.rfc-editor.org/>.) See  
22 System.Xml.XmlTextWriter.WriteBase64(System.Byte[],System.Int32,Syste  
23 m.Int32) (in the XmlTextWriter class) for an example using this method. Byte  
24 array to encode. The position within the buffer indicating the start of the  
25 bytes to write. The number of bytes to write.



## WriteBinHex

**[C#] public abstract void WriteBinHex(byte[] buffer, int index, int count);**  
**[C++] public: virtual void WriteBinHex(unsigned char buffer \_\_gc[], int index, int count) = 0;**  
**[VB] MustOverride Public Sub WriteBinHex(ByVal buffer() As Byte, ByVal index As Integer, ByVal count As Integer)**  
**[JScript] public abstract function WriteBinHex(buffer : Byte[], index : int, count : int);**

### *Description*

When overridden in a derived class, encodes the specified binary bytes as binhex and writes out the resulting text. Byte array to encode. The position within the buffer indicating the start of the bytes to write. The number of bytes to write.

## WriteCData

**[C#] public abstract void WriteCData(string text);**  
**[C++] public: virtual void WriteCData(String\* text) = 0;**  
**[VB] MustOverride Public Sub WriteCData(ByVal text As String)**  
**[JScript] public abstract function WriteCData(text : String);**

### *Description*

When overridden in a derived class, writes out a block containing the specified text.

If *text* is either null or `String.Empty`, this method writes an empty CDATA block, for example `<![CDATA[ ]>`. The text to place inside the CDATA block.

`WriteCharEntity`

**[C#] public abstract void WriteCharEntity(char ch);**

**[C++] public: virtual void WriteCharEntity(\_\_wchar\_t ch) = 0;**

**[VB] MustOverride Public Sub WriteCharEntity(ByVal ch As Char)**

**[JScript] public abstract function WriteCharEntity(ch : Char);**

### ***Description***

When overridden in a derived class, forces the generation of a character entity for the specified Unicode character value.

This method writes the Unicode character in hexadecimal character entity reference format. The Unicode character for which to generate a character entity.

`WriteChars`

**[C#] public abstract void WriteChars(char[] buffer, int index, int count);**

**[C++] public: virtual void WriteChars(\_\_wchar\_t buffer \_\_gc[], int index, int count) = 0;**

**[VB] MustOverride Public Sub WriteChars(ByVal buffer() As Char, ByVal index As Integer, ByVal count As Integer)**

**[JScript] public abstract function WriteChars(buffer : Char[], index : int, count : int);**

## **Description**

When overridden in a derived class, writes text a buffer at a time.

This method can be used to write large amounts of text a buffer at a time. Character array containing the text to write. The position within the buffer indicating the start of the text to write. The number of characters to write.

WriteComment

[C#] public abstract void WriteComment(string text);

[C++] public: virtual void WriteComment(String\* text) = 0;

[VB] MustOverride Public Sub WriteComment(ByVal text As String)

[JScript] public abstract function WriteComment(text : String);

## **Description**

When overridden in a derived class, writes out a comment containing the specified text.

If *text* is either null or String.Empty, this method writes a Comment with no data content, for example . Text to place inside the comment.

WriteDocType

[C#] public abstract void WriteDocType(string name, string pubid, string sysid, string subset);

[C++] public: virtual void WriteDocType(String\* name, String\* pubid, String\* sysid, String\* subset) = 0;

1 [VB] MustOverride Public Sub WriteDocType(ByVal name As String, ByVal  
2 pubid As String, ByVal sysid As String, ByVal subset As String)

3 [JScript] public abstract function WriteDocType(name : String, pubid :  
4 String, sysid : String, subset : String);

5  
6 **Description**

7 When overridden in a derived class, writes the DOCTYPE declaration  
8 with the specified name and optional attributes.

9 This method does not check for invalid characters in *pubid*, *sysid* or  
10 *subset* . The name of the DOCTYPE. This must be non-empty. If non-null it  
11 also writes PUBLIC "pubid" "sysid" where pubid and sysid are replaced  
12 with the value of the given arguments. If pubid is null and sysid is non-null it  
13 writes SYSTEM "sysid" where sysid is replaced with the value of this  
14 argument. If non-null it writes [subset] where subset is replaced with the  
15 value of this argument.

16 WriteElementString

17  
18 [C#] public void WriteElementString(string localName, string value);

19 [C++] public: void WriteElementString(String\* localName, String\* value);

20 [VB] Public Sub WriteElementString(ByVal localName As String, ByVal  
21 value As String)

22 [JScript] public function WriteElementString(localName : String, value :  
23 String); When overridden in a derived class, writes an element containing a  
24 string value.

## ***Description***

When overridden in a derived class, writes an element with the specified local name and value. The local name of the element The value of the element

WriteElementString

**[C#] public void WriteElementString(string localName, string ns, string value);**

**[C++] public: void WriteElementString(String\* localName, String\* ns, String\* value);**

**[VB] Public Sub WriteElementString(ByVal localName As String, ByVal ns As String, ByVal value As String)**

**[JScript] public function WriteElementString(localName : String, ns : String, value : String);**

## ***Description***

When overridden in a derived class, writes an element with the specified parameters. The local name of the element The namespace URI to associate with the element The value of the element

WriteEndAttribute

**[C#] public abstract void WriteEndAttribute();**

**[C++] public: virtual void WriteEndAttribute() = 0;**

**[VB] MustOverride Public Sub WriteEndAttribute()**

1 **[JScript] public abstract function WriteEndAttribute();**

3 ***Description***

4       **When overridden in a derived class, closes the previous**  
5 **System.Xml.XmlWriter.WriteStartAttribute(System.String,System.String)**  
6 **call.**

7       **If you call WriteStartAttribute , you can close the attribute with this**  
8 **method.**

9       WriteEndDocument

11 **[C#] public abstract void WriteEndDocument();**

12 **[C++] public: virtual void WriteEndDocument() = 0;**

13 **[VB] MustOverride Public Sub WriteEndDocument()**

14 **[JScript] public abstract function WriteEndDocument();**

16 ***Description***

17       **When overridden in a derived class, closes any open elements or**  
18 **attributes and puts the writer back in the Start state.**

19       WriteEndElement

21 **[C#] public abstract void WriteEndElement();**

22 **[C++] public: virtual void WriteEndElement() = 0;**

23 **[VB] MustOverride Public Sub WriteEndElement()**

24 **[JScript] public abstract function WriteEndElement();**

## **Description**

When overridden in a derived class, closes one element and pops the corresponding namespace scope.

If the element contains no content, a short end tag ">" is written; otherwise, a full end tag is written.

WriteEntityRef

[C#] public abstract void WriteEntityRef(string name);

[C++] public: virtual void WriteEntityRef(String\* name) = 0;

[VB] MustOverride Public Sub WriteEntityRef(ByVal name As String)

[JScript] public abstract function WriteEntityRef(name : String);

## **Description**

When overridden in a derived class, writes out an entity reference as follows: & name ;. The name of the entity reference.

WriteFullEndElement

[C#] public abstract void WriteFullEndElement();

[C++] public: virtual void WriteFullEndElement() = 0;

[VB] MustOverride Public Sub WriteFullEndElement()

[JScript] public abstract function WriteFullEndElement();

## **Description**

1       When overridden in a derived class, closes one element and pops the  
2 corresponding namespace scope.

3       This method always writes the full end tag. This is useful when dealing  
4 with elements that must include a full end tag. For example, browsers expect  
5 HTML script blocks to be closed with "".

6       WriteName

7  
8 [C#] public abstract void WriteName(string name);

9 [C++] public: virtual void WriteName(String\* name) = 0;

10 [VB] MustOverride Public Sub WriteName(ByVal name As String)

11 [JScript] public abstract function WriteName(name : String);

12  
13 *Description*

14       When overridden in a derived class, writes out the specified name,  
15 ensuring it is a valid name according to the W3C XML 1.0 recommendation  
16 (<http://www.w3.org/TR/1998/REC-xml-19980210#NT-Name>).

17       If System.Xml.XmlTextWriter.Namespaces is set to true , WriteName  
18 also checks that the name is also valid according to the W3C Namespaces in  
19 XML recommendation. The name to write.

20       WriteNmToken

21  
22 [C#] public abstract void WriteNmToken(string name);

23 [C++] public: virtual void WriteNmToken(String\* name) = 0;

24 [VB] MustOverride Public Sub WriteNmToken(ByVal name As String)

25 [JScript] public abstract function WriteNmToken(name : String);



## **Description**

When overridden in a derived class, writes out the specified name, ensuring it is a valid NmToken according to the W3C XML 1.0 recommendation (<http://www.w3.org/TR/1998/REC-xml-19980210#NT-Name>). The name to write.

WriteNode

**[C#] public virtual void WriteNode(XmlReader reader, bool defattr);**

**[C++] public: virtual void WriteNode(XmlReader\* reader, bool defattr);**

**[VB] Overridable Public Sub WriteNode(ByVal reader As XmlReader, ByVal defattr As Boolean)**

**[JScript] public function WriteNode(reader : XmlReader, defattr : Boolean);**

## **Description**

When overridden in a derived class, copies everything from the reader to the writer and moves the reader to the start of the next sibling.

The following table shows the supported node types for this method. The System.Xml.XmlReader to read from. true to copy the default attributes from the XmlReader ; otherwise, false.

WriteProcessingInstruction

**[C#] public abstract void WriteProcessingInstruction(string name, string text);**

**[C++] public: virtual void WriteProcessingInstruction(String\* name, String\***

1 **text) = 0;**

2 **[VB] MustOverride Public Sub WriteProcessingInstruction(ByVal name As**  
3 **String, ByVal text As String)**

4 **[JScript] public abstract function WriteProcessingInstruction(name : String,**  
5 **text : String);**

6  
7 ***Description***

8 **When overridden in a derived class, writes out a processing instruction**  
9 **with a space between the name and text as follows: .**

10 **This method can be used to write the XML declaration (rather than**  
11 **System.Xml.XmlWriter.WriteStartDocument ). This could result in the**  
12 **encoding attribute being incorrectly written. For example, the following C#**  
13 **code would result in an invalid XML document. The name of the processing**  
14 **instruction. The text to include in the processing instruction.**

15 **WriteQualifiedName**

16  
17 **[C#] public abstract void WriteQualifiedName(string localName, string ns);**

18 **[C++] public: virtual void WriteQualifiedName(String\* localName, String\***  
19 **ns) = 0;**

20 **[VB] MustOverride Public Sub WriteQualifiedName(ByVal localName As**  
21 **String, ByVal ns As String)**

22 **[JScript] public abstract function WriteQualifiedName(localName : String,**  
23 **ns : String);**

24  
25 ***Description***

When overridden in a derived class, writes out the namespace-qualified name. This method looks up the prefix that is in scope for the given namespace.

For example, the following C# code: `writer.Formatting = Formatting.Indented; writer.WriteStartElement("root"); writer.WriteAttributeString("xmlns", "x", null, "urn:abc"); writer.WriteStartElement("item"); writer.WriteStartAttribute("href", null); writer.WriteString("#"); writer.WriteQualifiedName("test", "urn:abc"); writer.WriteEndAttribute(); writer.WriteEndElement(); writer.WriteEndElement(); writer.Close();` Generates the following output: If *ns* maps to the current default namespace, no prefix is generated. The local name to write. The namespace URI for the name.

WriteRaw

[C#] public abstract void WriteRaw(string data);

[C++] public: virtual void WriteRaw(String\* data) = 0;

[VB] MustOverride Public Sub WriteRaw(ByVal data As String)

[JScript] public abstract function WriteRaw(data : String);

### *Description*

When overridden in a derived class, writes raw markup manually from a string.

This method bypasses the entitization of special characters. String containing the text to write.

WriteRaw

1  
2 **[C#] public abstract void WriteRaw(char[] buffer, int index, int count);**  
3 **[C++] public: virtual void WriteRaw(\_\_wchar\_t buffer \_\_gc[], int index, int**  
4 **count) = 0;**  
5 **[VB] MustOverride Public Sub WriteRaw(ByVal buffer() As Char, ByVal**  
6 **index As Integer, ByVal count As Integer)**  
7 **[JScript] public abstract function WriteRaw(buffer : Char[], index : int,**  
8 **count : int); When overridden in a derived class, writes raw markup**  
9 **manually.**

### 11 *Description*

12 **When overridden in a derived class, writes raw markup manually from**  
13 **a character buffer.**

14 **This method bypasses the entitization of special characters. Character**  
15 **array containing the text to write. The position within the buffer indicating**  
16 **the start of the text to write. The number of characters to write.**

17 **WriteStartAttribute**

18  
19 **[C#] public void WriteStartAttribute(string localName, string ns);**  
20 **[C++] public: void WriteStartAttribute(String\* localName, String\* ns);**  
21 **[VB] Public Sub WriteStartAttribute(ByVal localName As String, ByVal ns**  
22 **As String)**  
23 **[JScript] public function WriteStartAttribute(localName : String, ns :**  
24 **String); When overridden in a derived class, writes the start of an attribute.**  
25

## **Description**

When overridden in a derived class, writes the start of an attribute.

This is a more advanced version of

`System.Xml.XmlWriter.WriteString(System.String, System.String, System.String)` that allows you to write an attribute value using multiple write methods, such as `System.Xml.XmlWriter.WriteString(System.String)`, `System.Xml.XmlWriter.WriteQualifiedName(System.String, System.String)`, and so on. The local name of the attribute. The namespace URI of the attribute

`WriteStartAttribute`

**[C#]** `public abstract void WriteStartAttribute(string prefix, string localName, string ns);`

**[C++]** `public: virtual void WriteStartAttribute(String* prefix, String* localName, String* ns) = 0;`

**[VB]** `MustOverride Public Sub WriteStartAttribute(ByVal prefix As String, ByVal localName As String, ByVal ns As String)`

**[JScript]** `public abstract function WriteStartAttribute(prefix : String, localName : String, ns : String);`

## **Description**

When overridden in a derived class, writes the start of an attribute.

This method allows you to write a value using multiple Write methods.  
The namespace prefix of the attribute. The local name of the attribute. The  
namespace URI for the attribute.

WriteStartDocument

[C#] public abstract void WriteStartDocument();

[C++] public: virtual void WriteStartDocument() = 0;

[VB] MustOverride Public Sub WriteStartDocument()

[JScript] public abstract function WriteStartDocument(); When overridden  
in a derived class, writes the XML declaration.

### *Description*

When overridden in a derived class, writes the XML declaration with  
the version "1.0".

The encoding level of the document is determined by how the writer is  
implemented. For example, if an System.Text.Encoding object is specified in  
the XmlTextWriter constructor, this determines the value of the encoding  
attribute. This method does not create a standalone attribute.

WriteStartDocument

[C#] public abstract void WriteStartDocument(bool standalone);

[C++] public: virtual void WriteStartDocument(bool standalone) = 0;

[VB] MustOverride Public Sub WriteStartDocument(ByVal standalone As  
Boolean)

[JScript] public abstract function WriteStartDocument(standalone :

Boolean);

### *Description*

When overridden in a derived class, writes the XML declaration with the version "1.0" and the standalone attribute.

The encoding level of the document is determined by how the writer is implemented. For example, if an `System.Text.Encoding` object is specified in the `XmlTextWriter` constructor, this determines the value of the encoding attribute. If true, it writes "standalone=yes"; if false, it writes

"standalone=no"

`WriteStartElement`

[C#] `public void WriteStartElement(string localName);`

[C++] `public: void WriteStartElement(String* localName);`

[VB] `Public Sub WriteStartElement(ByVal localName As String)`

[JScript] `public function WriteStartElement(localName : String);`

### *Description*

When overridden in a derived class, writes out a start tag with the specified local name. The local name of the element.

`WriteStartElement`

[C#] `public void WriteStartElement(string localName, string ns);`

[C++] `public: void WriteStartElement(String* localName, String* ns);`

[VB] `Public Sub WriteStartElement(ByVal localName As String, ByVal ns As`

String)

[JScript] public function WriteStartElement(localName : String, ns : String);

When overridden in a derived class, writes the specified start tag.

### *Description*

When overridden in a derived class, writes the specified start tag and associates it with the given namespace.

After calling this method you can either write attributes, or create content using System.Xml.XmlWriter.WriteComment(System.String) , System.Xml.XmlWriter.WriteString(System.String) , or WriteStartElement for child elements. You must close this element with either

System.Xml.XmlWriter.WriteEndElement or

System.Xml.XmlWriter.WriteFullEndElement . For example, the following

C# code writer.WriteStartElement("item",null); writer.WriteString("some text"); writer.WriteEndElement(); generates the following output: some text

See

System.Xml.XmlTextWriter.WriteStartElement(System.String,System.String ,System.String) (in the XmlTextWriter class) for an example using this

method. The local name of the element. The namespace URI to associate with the element. If this namespace is already in scope and has an associated prefix, the writer automatically writes that prefix also.

WriteStartElement

[C#] public abstract void WriteStartElement(string prefix, string localName, string ns);



```

1  [C++] public: virtual void WriteStartElement(String* prefix, String*
2  localName, String* ns) = 0;
3  [VB] MustOverride Public Sub WriteStartElement(ByVal prefix As String,
4  ByVal localName As String, ByVal ns As String)
5  [JScript] public abstract function WriteStartElement(prefix : String,
6  localName : String, ns : String);

```

### *Description*

When overridden in a derived class, writes the specified start tag and associates it with the given namespace and prefix. The namespace prefix of the element. The local name of the element. The namespace URI to associate with the element.

WriteString

```

15 [C#] public abstract void WriteString(string text);
16 [C++] public: virtual void WriteString(String* text) = 0;
17 [VB] MustOverride Public Sub WriteString(ByVal text As String)
18 [JScript] public abstract function WriteString(text : String);

```

### *Description*

When overridden in a derived class, writes the given text content.

WriteString does the following The characters & , < , and > are replaced with & , < , and > , respectively. The text to write.

WriteSurrogateCharEntity

```

1  [C#] public abstract void WriteSurrogateCharEntity(char lowChar, char
2  highChar);
3
4  [C++] public: virtual void WriteSurrogateCharEntity(__wchar_t lowChar,
5  __wchar_t highChar) = 0;
6
7  [VB] MustOverride Public Sub WriteSurrogateCharEntity(ByVal lowChar
8  As Char, ByVal highChar As Char)
9
10 [JScript] public abstract function WriteSurrogateCharEntity(lowChar :
11 Char, highChar : Char);

```

### ***Description***

When overridden in a derived class, generates and writes the surrogate character entity for the surrogate character pair.

The surrogate character entity is written in hexadecimal format. The range for surrogate characters is #x10000 to #x10FFFF. The following formula is used to generate the surrogate character entity: (*highChar* - 0xD800) \* 0x400 + (*lowChar* - 0xDC00) + 0x10000 Applications encode DOM strings using UTF-16. For both HTML and XML, the document character set (and therefore the notation of numeric character references) is based on UCS [ISO-10646]. A single numeric character reference in a source document may therefore in some cases correspond to two 16-bit units in a DOM string (a high surrogate and a low surrogate). These 16-bit units are referred to as a surrogate pair. The low surrogate. This must be a value between 0xDC00 and 0xDFFF. The high surrogate. This must be a value between 0xD800 and 0xDBFF.

WriteWhitespace

**[C#] public abstract void WriteWhitespace(string ws);**

**[C++] public: virtual void WriteWhitespace(String\* ws) = 0;**

**[VB] MustOverride Public Sub WriteWhitespace(ByVal ws As String)**

**[JScript] public abstract function WriteWhitespace(ws : String);**

### **System.Xml.Schema**

This is the namespace for the XML classes that provide standards-based support for XML schemas. The supported standards are: XML Schemas for Structures - <http://www.w3.org/TR/xmlschema-1/> - supports for schema mapping and for validation. See also XmlSchemaCollection which currently provides XSD and XDR schema validation.

#### *Description*

This is the namespace for the XML classes that provide standards-based support for XML schemas. The supported standards are: XML Schemas for Structures - <http://www.w3.org/TR/xmlschema-1/> - supports for schema mapping and for validation. See also XmlSchemaCollection which currently provides XSD and XDR schema validation.

ValidationEventArgs class (System.Xml.Schema)

#### *Description*

Returns detailed information relating to the **ValidationEventHandler** .

Properties:

Exception

[C#] public XmlSchemaException Exception {get;}

[C++] public: \_\_property XmlSchemaException\* get\_Exception();

[VB] Public ReadOnly Property Exception As XmlSchemaException

[JScript] public function get Exception() : XmlSchemaException;

### *Description*

Gets the **System.Xml.Schema.XmlSchemaException** associated with the validation event.

Message

[C#] public string Message {get;}

[C++] public: \_\_property String\* get\_Message();

[VB] Public ReadOnly Property Message As String

[JScript] public function get Message() : String;

### *Description*

Gets the text description corresponding to the validation event.

Severity

[C#] public XmlSeverityType Severity {get;}

[C++] public: \_\_property XmlSeverityType get\_Severity();

1 [VB] Public ReadOnly Property Severity As XmlSeverityType

2 [JScript] public function get Severity() : XmlSeverityType;

3  
4 *Description*

5 Gets the severity of the validation event.

6 Methods:

7 ValidationEventHandler delegate (System.Xml.Schema)

8 ToString

9  
10  
11 *Description*

12 Represents the method that will handle the

13 **System.Xml.Schema.ValidationEventArgs** .

14 XmlSchema class (System.Xml.Schema)

15 ToString

16  
17  
18 *Description*

19 Contains the definition of a schema. The class for **schema** element. All  
20 XML Schema Definition language (XSD) elements are children of the **schema**  
21 element. Represents the W3C **schema** element.

22 ToString

23  
24 [C#] public const string InstanceNamespace;

25 [C++] public: const String\* InstanceNamespace;

1 [VB] Public Const InstanceNamespace As String

2 [JScript] public var InstanceNamespace : String;

3

4 *Description*

5 Provides the instance namespace.

6 ToString

7

8 [C#] public const string Namespace;

9 [C++] public: const String\* Namespace;

10 [VB] Public Const Namespace As String

11 [JScript] public var Namespace : String;

12

13 *Description*

14 Provides the namespace for validation.

15 Constructors:

16 XmlSchema

17 *Example Syntax:*

18 ToString

19

20 [C#] public XmlSchema();

21 [C++] public: XmlSchema();

22 [VB] Public Sub New()

23 [JScript] public function XmlSchema();

24

25 *Description*

Constructs a new, empty schema.

Annotation

AttributeFormDefault

ToString

### Description

Indicates the form for attributes declared in the target namespace of the schema.

The value must be one of **XmlSchemaForm** values. The default value is **unqualified**.

AttributeGroups

ToString

[C#] public XmlSchemaObjectTable AttributeGroups {get;}

[C++] public: \_\_property XmlSchemaObjectTable\* get\_AttributeGroups();

[VB] Public ReadOnly Property AttributeGroups As XmlSchemaObjectTable

[JScript] public function get AttributeGroups() : XmlSchemaObjectTable;

### Description

The **XmlSchemaObjectTable** for all attribute groups in the schema.

Attributes

ToString

[C#] public XmlSchemaObjectTable Attributes {get;}

```

1 [C++] public: __property XmlSchemaObjectTable* get_Attributes();
2 [VB] Public ReadOnly Property Attributes As XmlSchemaObjectTable
3 [JScript] public function get Attributes() : XmlSchemaObjectTable;

```

#### *Description*

The **XmlSchemaObjectTable** for all attributes in the schema.

BlockDefault

ToString

```

10 [C#] public XmlSchemaDerivationMethod BlockDefault {get; set;}
11 [C++] public: __property XmlSchemaDerivationMethod
12 get_BlockDefault();public: __property void
13 set_BlockDefault(XmlSchemaDerivationMethod);
14 [VB] Public Property BlockDefault As XmlSchemaDerivationMethod
15 [JScript] public function get BlockDefault() :
16 XmlSchemaDerivationMethod;public function set
17 BlockDefault(XmlSchemaDerivationMethod);

```

#### *Description*

Type of derivation. The **BlockDefault** attribute sets the default value of the **block** attribute on element and complex type elements in the target namespace of the schema. The **block** attribute prevents a complex type (or element) that has the specified type of derivation from being used in place of the inherited complex type (or element).



This value can contain **all** or a list that is a subset of extension/restriction/substitution.

ElementFormDefault

ToString

[C#] public XmlSchemaForm ElementFormDefault {get; set;}

[C++] public: \_\_property XmlSchemaForm get\_ElementFormDefault();public: \_\_property void set\_ElementFormDefault(XmlSchemaForm);

[VB] Public Property ElementFormDefault As XmlSchemaForm

[JScript] public function get ElementFormDefault() : XmlSchemaForm;public function set ElementFormDefault(XmlSchemaForm);

### *Description*

Indicates the form for elements declared in the target namespace of the schema.

The value must be one of the **XmlSchemaForm** values. The default value is **Unqualified** .

Elements

ToString

[C#] public XmlSchemaObjectTable Elements {get;}

[C++] public: \_\_property XmlSchemaObjectTable\* get\_Elements();

[VB] Public ReadOnly Property Elements As XmlSchemaObjectTable

[JScript] public function get Elements() : XmlSchemaObjectTable;

## Description

The **XmlSchemaObjectTable** for all elements in the schema.

FinalDefault

ToString

[C#] public XmlSchemaDerivationMethod FinalDefault {get; set;}

[C++] public: \_\_property XmlSchemaDerivationMethod

get\_FinalDefault();public: \_\_property void

set\_FinalDefault(XmlSchemaDerivationMethod);

[VB] Public Property FinalDefault As XmlSchemaDerivationMethod

[JScript] public function get FinalDefault() : XmlSchemaDerivationMethod;public

function set FinalDefault(XmlSchemaDerivationMethod);

## Description

Type of derivation. The **FinalDefault** attribute sets the default value of the **final** attribute on element and complex type elements in the target namespace of this schema. The **final** attribute prevents the specified type of derivation of a element or complex type.

This value can contain **all** or a list that is a subset of extension/restriction.

Groups

ToString

[C#] public XmlSchemaObjectTable Groups {get;}

[C++] public: \_\_property XmlSchemaObjectTable\* get\_Groups();

[VB] Public ReadOnly Property Groups As XmlSchemaObjectTable

[JScript] public function get Groups() : XmlSchemaObjectTable;

#### *Description*

The **XmlSchemaObjectTable** for all groups in the schema.

Id

Includes

ToString

#### *Description*

Collection of included and imported schemas.

IsCompiled

ToString

[C#] public bool IsCompiled {get;}

[C++] public: \_\_property bool get\_IsCompiled();

[VB] Public ReadOnly Property IsCompiled As Boolean

[JScript] public function get IsCompiled() : Boolean;

#### *Description*

Indicates if the schema has been compiled.

If value is **true** , the schema has been compiled.

Items

ToString

1  
2 [C#] public XmlSchemaObjectCollection Items {get;}

3 [C++] public: \_\_property XmlSchemaObjectCollection\* get\_Items();

4 [VB] Public ReadOnly Property Items As XmlSchemaObjectCollection

5 [JScript] public function get Items() : XmlSchemaObjectCollection;

6  
7 *Description*

8 This **Items** collection is used to add new element types at the **schema**  
9 element level.

10 Collection of **XmlSchemaAnnotation** , **XmlSchemaAttribute** ,  
11 **XmlSchemaAttributeGroup** , **XmlSchemaComplexType** ,  
12 **XmlSchemaSimpleType** , **XmlSchemaElement** , **XmlSchemaGroup** , or  
13 **XmlSchemaNotation** .

14 LineNumber

15 LinePosition

16 Notations

17 ToString

18  
19  
20 *Description*

21 The **XmlSchemaObjectTable** for all notations in the schema.

22 SchemaTypes

23 ToString

24  
25 [C#] public XmlSchemaObjectTable SchemaTypes {get;}

```
1 [C++] public: __property XmlSchemaObjectTable* get_SchemaTypes();
2 [VB] Public ReadOnly Property SchemaTypes As XmlSchemaObjectTable
3 [JScript] public function get SchemaTypes() : XmlSchemaObjectTable;
```

#### Description

The **XmlSchemaObjectTable** for all schema types in the schema.

SourceUri

TargetNamespace

ToString

#### Description

Property for the schema target namespace. The URI reference of the namespace of this schema.

This is the namespace of all schema components in this schema as well as any schema included using the **include** element. Included schemas must either have the same target namespace as the containing schema or have no target namespace specified.

UnhandledAttributes

Version

ToString

#### Description

Version of the schema.

Optional.

Compile

[C#] public void Compile(ValidationEventHandler validationEventHandler);

[C++] public: void Compile(ValidationEventHandler\* validationEventHandler);

[VB] Public Sub Compile(ByVal validationEventHandler As

ValidationEventHandler)

[JScript] public function Compile(validationEventHandler :

ValidationEventHandler);

### *Description*

Compiles the XML Schema Definition language (XSD) Schema Object Model (SOM) into schema information for validation. Used to check the syntactic and semantic structure of the programmatically built SOM. Semantic validation checking is performed before compiling. Validation Event Handler

Read

[C#] public static XmlSchema Read(Stream stream, ValidationEventHandler validationEventHandler);

[C++] public: static XmlSchema\* Read(Stream\* stream, ValidationEventHandler\* validationEventHandler);

[VB] Public Shared Function Read(ByVal stream As Stream, ByVal validationEventHandler As ValidationEventHandler) As XmlSchema

[JScript] public static function Read(stream : Stream, validationEventHandler : ValidationEventHandler) : XmlSchema;

## Description

Reads an XML Schema Definition language (XSD) schema from the supplied stream. Data stream Validation Event Handler

Read

[C#] public static XmlSchema Read(TextReader reader, ValidationEventHandler validationEventHandler);

[C++] public: static XmlSchema\* Read(TextReader\* reader, ValidationEventHandler\* validationEventHandler);

[VB] Public Shared Function Read(ByVal reader As TextReader, ByVal validationEventHandler As ValidationEventHandler) As XmlSchema

[JScript] public static function Read(reader : TextReader, validationEventHandler : ValidationEventHandler) : XmlSchema; Reads an XML Schema Definition language (XSD) schema.

## Description

Reads an XSD schema from the supplied text reader. TextReader Validation Event Handler

Read

[C#] public static XmlSchema Read(XmlReader reader, ValidationEventHandler validationEventHandler);

[C++] public: static XmlSchema\* Read(XmlReader\* reader, ValidationEventHandler\* validationEventHandler);

```

1 [VB] Public Shared Function Read(ByVal reader As XmlReader, ByVal
2 validationEventHandler As ValidationEventHandler) As XmlSchema
3 [JScript] public static function Read(reader : XmlReader, validationEventHandler
4 : ValidationEventHandler) : XmlSchema;

```

### *Description*

Reads an XML Schema Definition language (XSD) schema from the supplied reader. XmlReader Validation Event Handler

### Write

```

11 [C#] public void Write(Stream stream);
12 [C++] public: void Write(Stream* stream);
13 [VB] Public Sub Write(ByVal stream As Stream)
14 [JScript] public function Write(stream : Stream); Writes out an XML Schema
15 Definition language (XSD) schema.

```

### *Description*

Writes the XSD schema to the supplied data stream. Data stream

### Write

```

21 [C#] public void Write(TextWriter writer);
22 [C++] public: void Write(TextWriter* writer);
23 [VB] Public Sub Write(ByVal writer As TextWriter)
24 [JScript] public function Write(writer : TextWriter);

```



## Description

Writes the XML Schema Definition language (XSD) schema to the supplied **XmlWriter** . XmlWriter

Write

[C#] public void Write(XmlWriter writer);

[C++] public: void Write(XmlWriter\* writer);

[VB] Public Sub Write(ByVal writer As XmlWriter)

[JScript] public function Write(writer : XmlWriter);

## Description

Writes out the XML Schema Definition language (XSD) schema.

XmlWriter

Write

[C#] public void Write(Stream stream, XmlNamespaceManager namespaceManager);

[C++] public: void Write(Stream\* stream, XmlNamespaceManager\* namespaceManager);

[VB] Public Sub Write(ByVal stream As Stream, ByVal namespaceManager As XmlNamespaceManager)

[JScript] public function Write(stream : Stream, namespaceManager : XmlNamespaceManager);

*Description*

Writes the XML Schema Definition language (XSD) schema to the supplied **TextWriter** .

Write

[C#] public void Write(TextWriter writer, XmlNamespaceManager namespaceManager);

[C++] public: void Write(TextWriter\* writer, XmlNamespaceManager\* namespaceManager);

[VB] Public Sub Write(ByVal writer As TextWriter, ByVal namespaceManager As XmlNamespaceManager)

[JScript] public function Write(writer : TextWriter, namespaceManager : XmlNamespaceManager);

*Description*

Writes the XML Schema Definition language (XSD) schema to the supplied **XmlWriter** . TextWriter XmlNamespaceManager

Write

[C#] public void Write(XmlWriter writer, XmlNamespaceManager namespaceManager);

[C++] public: void Write(XmlWriter\* writer, XmlNamespaceManager\* namespaceManager);

[VB] Public Sub Write(ByVal writer As XmlWriter, ByVal namespaceManager

1 As XmlNamespaceManager)

2 [JScript] public function Write(writer : XmlWriter, namespaceManager :  
3 XmlNamespaceManager);

4  
5 *Description*

6 Writes out the XML Schema Definition language (XSD) schema.

7 XmlWriter XmlNamespaceManager

8 XmlSchemaAll class (System.Xml.Schema)

9 Write

10  
11  
12 *Description*

13 Permits the elements in the group to appear in any order in the containing  
14 element. Represents the W3C **all** element (compositor).

15 XmlSchemaAll

16 *Example Syntax:*

17 Write

18  
19 [C#] public XmlSchemaAll();

20 [C++] public: XmlSchemaAll();

21 [VB] Public Sub New()

22 [JScript] public function XmlSchemaAll();

23 Annotation

24 Id

25 Items

Write

*Description*

Collection of **XmlSchemaElement** , elements contained within the **all** compositor.

LineNumber

LinePosition

MaxOccurs

MaxOccursString

MinOccurs

MinOccursString

SourceUri

UnhandledAttributes

XmlSchemaAnnotated class (System.Xml.Schema)

ToString

*Description*

Class for all element types. The class for any element that can contain **annotation** elements.

XmlSchemaAnnotated

*Example Syntax:*

ToString

[C#] public XmlSchemaAnnotated();

[C++] public: XmlSchemaAnnotated();

[VB] Public Sub New()

[JScript] public function XmlSchemaAnnotated();

Annotation

ToString

[C#] public XmlSchemaAnnotation Annotation {get; set;}

[C++] public: \_\_property XmlSchemaAnnotation\* get\_Annotation();public:

\_\_property void set\_Annotation(XmlSchemaAnnotation\*);

[VB] Public Property Annotation As XmlSchemaAnnotation

[JScript] public function get Annotation() : XmlSchemaAnnotation;public

function set Annotation(XmlSchemaAnnotation);

### *Description*

Sets or retrieves the **annotation** property.

Id

ToString

[C#] public string Id {get; set;}

[C++] public: \_\_property String\* get\_Id();public: \_\_property void set\_Id(String\*);

[VB] Public Property Id As String

[JScript] public function get Id() : String;public function set Id(String);

1  
2 *Description*

3 Provides the string id.

4 The id value must be of type ID and be unique within the document  
5 containing this element.

6 LineNumber

7 LinePosition

8 SourceUri

9 UnhandledAttributes

10 ToString

11  
12  
13 *Description*

14 Stores qualified attributes that do not belong to the schema target  
15 namespace.

16 XmlSchemaAnnotation class (System.Xml.Schema)

17 ToString

18  
19  
20 *Description*

21 Defines an annotation. Creates an **annotation** element. Represents the  
22 W3C **annotation** element.

23 An **annotation** element can contain one or more **XmlSchemaAppInfo**  
24 classes (information for applications) and **XmlSchemaDocumentation** classes  
25 (comments or text for humans).

XmlSchemaAnnotation

*Example Syntax:*

ToString

[C#] public XmlSchemaAnnotation();

[C++] public: XmlSchemaAnnotation();

[VB] Public Sub New()

[JScript] public function XmlSchemaAnnotation();

Items

ToString

[C#] public XmlSchemaObjectCollection Items {get;}

[C++] public: \_\_property XmlSchemaObjectCollection\* get\_Items();

[VB] Public ReadOnly Property Items As XmlSchemaObjectCollection

[JScript] public function get Items() : XmlSchemaObjectCollection;

*Description*

Provides a collection of **annotation** elements.

Read-only property.

LineNumber

LinePosition

SourceUri

XmlSchemaAny class (System.Xml.Schema)

ToString

*Description*

Enables any element from the specified namespace(s) to appear in the containing **complexType** , **sequence** , or **choice** element. Represents the W3C **any** element.

XmlSchemaAny

*Example Syntax:*

ToString

[C#] public XmlSchemaAny();

[C++] public: XmlSchemaAny();

[VB] Public Sub New()

[JScript] public function XmlSchemaAny();

Annotation

Id

LineNumber

LinePosition

MaxOccurs

MaxOccursString

MinOccurs

MinOccursString

Namespace

ToString



1  
2  
3 *Description*

4 Namespaces containing the elements that can be used.

5 If no namespace is specified, **any** is the default. If the namespace is  
6 specified, it must be one of the following: Namespace Value Description **any**

7 Elements from any namespace can be present.

8 ProcessContents

9 ToString

10  
11 [C#] public XmlSchemaContentProcessing ProcessContents {get; set;}

12 [C++] public: \_\_property XmlSchemaContentProcessing

13 get\_ProcessContents();public: \_\_property void

14 set\_ProcessContents(XmlSchemaContentProcessing);

15 [VB] Public Property ProcessContents As XmlSchemaContentProcessing

16 [JScript] public function get ProcessContents() :

17 XmlSchemaContentProcessing;public function set

18 ProcessContents(XmlSchemaContentProcessing);

19  
20 *Description*

21 Indicates how an application or XML processor should handle the  
22 validation of XML documents for the elements specified by the **any** element.

23 If no **processContents** attribute is specified, the default is **strict** . If  
24 **processContents** is specified, it must one of the following: Enum Description

**Skip** The XML processor does not attempt to validate any attributes from the specified namespace.

SourceUri

UnhandledAttributes

XmlSchemaAnyAttribute class (System.Xml.Schema)

ToString

### *Description*

Enables any attribute from the specified namespace(s) to appear in the containing **complexType** element. Represents the W3C **anyAttribute** element.

XmlSchemaAnyAttribute

### *Example Syntax:*

ToString

[C#] public XmlSchemaAnyAttribute();

[C++] public: XmlSchemaAnyAttribute();

[VB] Public Sub New()

[JScript] public function XmlSchemaAnyAttribute();

Annotation

Id

LineNumber

LinePosition

Namespace

ToString

1  
2  
3 *Description*

4       Namespaces containing the attributes that can be used.

5       If no namespace is specified, **any** is the default. If the namespace is  
6 specified, it must one of the following. If **processContents** is specified, it must  
7 one of the following: Namespace Value Description **any** Elements from any  
8 namespace can be present.

9       ProcessContents

10      ToString

11  
12 [C#] public XmlSchemaContentProcessing ProcessContents {get; set;}

13 [C++] public: \_\_property XmlSchemaContentProcessing

14 get\_ProcessContents();public: \_\_property void

15 set\_ProcessContents(XmlSchemaContentProcessing);

16 [VB] Public Property ProcessContents As XmlSchemaContentProcessing

17 [JScript] public function get ProcessContents() :

18 XmlSchemaContentProcessing;public function set

19 ProcessContents(XmlSchemaContentProcessing);  
20

21 *Description*

22       Indicates how an application or XML processor should handle the  
23 validation of XML documents for the attributes specified by the **anyAttribute**  
24 element.  
25

If no **processContents** attribute is specified, the default is **strict** . If **processContents** is specified, it must one of the following.

SourceUri

UnhandledAttributes

XmlSchemaAppInfo class (System.Xml.Schema)

ToString

#### *Description*

Class that specifies information to be used by applications within an annotation. Represents the W3C **AppInfo** element.

XmlSchemaAppInfo

#### *Example Syntax:*

ToString

[C#] public XmlSchemaAppInfo();

[C++] public: XmlSchemaAppInfo();

[VB] Public Sub New()

[JScript] public function XmlSchemaAppInfo();

LineNumber

LinePosition

Markup

ToString

### Description

Returns an array of **XmlNode** that represents the document text markup.

Source

ToString

[C#] public string Source {get; set;}

[C++] public: \_\_property String\* get\_Source();public: \_\_property void

set\_Source(String\*);

[VB] Public Property Source As String

[JScript] public function get Source() : String;public function set Source(String);

### Description

Provides the source of the application information.

The source must be a URI reference. Optional.

SourceUri

XmlSchemaAttribute class (System.Xml.Schema)

ToString

### Description

Class for attribute types. Represents the W3C **attribute** element.

Attribute declarations can be present as child elements of the **schema** element (having global scope) or within complex type definitions. For complex

types, attribute declarations can be present as local declarations or references to attributes with global scope. Both global and local attribute declarations have the optional type attribute that refers to an existing simple type. If the attribute is not used, then the attribute (global or local) must define a local simple type.

XmlSchemaAttribute

*Example Syntax:*

ToString

[C#] public XmlSchemaAttribute();

[C++] public: XmlSchemaAttribute();

[VB] Public Sub New()

[JScript] public function XmlSchemaAttribute();

Annotation

AttributeType

ToString

### *Description*

Name of a built-in data type defined in this schema (or another schema indicated by the specified namespace).

The type must be a QName. The type can include a namespace prefix.

DefaultValue

ToString

[C#] public string DefaultValue {get; set;}

```

1 [C++] public: __property String* get_DefaultValue();public: __property void
2 set_DefaultValue(String*);
3 [VB] Public Property DefaultValue As String
4 [JScript] public function get DefaultValue() : String;public function set
5 DefaultValue(String);
6

```

### Description

The default value for the attribute.

The fixed and default properties are mutually exclusive.

FixedValue

ToString

```

13 [C#] public string FixedValue {get; set;}
14 [C++] public: __property String* get_FixedValue();public: __property void
15 set_FixedValue(String*);
16 [VB] Public Property FixedValue As String
17 [JScript] public function get FixedValue() : String;public function set
18 FixedValue(String);
19

```

### Description

The fixed value for the attribute.

This is a predetermined and unchangeable value.

Form

ToString

[C#] public XmlSchemaForm Form {get; set;}

[C++] public: \_\_property XmlSchemaForm get\_Form();public: \_\_property void  
set\_Form(XmlSchemaForm);

[VB] Public Property Form As XmlSchemaForm

[JScript] public function get Form() : XmlSchemaForm;public function set  
Form(XmlSchemaForm);

### *Description*

Form for the attribute.

The default value is the value of the **attributeFormDefault** attribute of the **schema** element containing the attribute. The value of **XmlSchemaForm** is defined as one of the following: Enum Description **Qualified** Attribute is not required to be qualified with the namespace prefix.

Id

LineNumber

LinePosition

Name

ToString

### *Description*

Provides the name of the attribute.

The name must be an NCName as defined in the W3C XML Namespaces specification. When a XML document is validated against a schema, each attribute



in the document is validated against an **attribute** element in the schema. The **attribute** element with a **name** attribute that matches the local name of the attribute in the document is used to validate that attribute.

QualifiedName

ToString

```
[C#] public XmlQualifiedName QualifiedName {get;}
```

```
[C++] public: __property XmlQualifiedName* get_QualifiedName();
```

```
[VB] Public ReadOnly Property QualifiedName As XmlQualifiedName
```

```
[JScript] public function get QualifiedName() : XmlQualifiedName;
```

### Description

The qualified name for the attribute.

Optional.

RefName

ToString

```
[C#] public XmlQualifiedName RefName {get; set;}
```

```
[C++] public: __property XmlQualifiedName* get_RefName();public: __property  
void set_RefName(XmlQualifiedName*);
```

```
[VB] Public Property RefName As XmlQualifiedName
```

```
[JScript] public function get RefName() : XmlQualifiedName;public function set  
RefName(XmlQualifiedName);
```

### Description

1       Name of an attribute declared (referenced) in this schema (or another  
2 schema indicated by the specified namespace).

3       The **RefName** must be a QName. The type can include a namespace prefix.

4       SchemaType

5       ToString

6  
7 [C#] public XmlSchemaSimpleType SchemaType {get; set;}

8 [C++] public: \_\_property XmlSchemaSimpleType\* get\_SchemaType();public:

9 \_\_property void set\_SchemaType(XmlSchemaSimpleType\*);

10 [VB] Public Property SchemaType As XmlSchemaSimpleType

11 [JScript] public function get SchemaType() : XmlSchemaSimpleType;public

12 function set SchemaType(XmlSchemaSimpleType);

13  
14 *Description*

15       Name of a built-in data type or simple type defined in this schema (or  
16 another schema indicated by the specified namespace).

17       The type must be a QName. The type can include a namespace prefix.

18       SchemaTypeName

19       ToString

20  
21 [C#] public XmlQualifiedName SchemaTypeName {get; set;}

22 [C++] public: \_\_property XmlQualifiedName\* get\_SchemaTypeName();public:

23 \_\_property void set\_SchemaTypeName(XmlQualifiedName\*);

24 [VB] Public Property SchemaTypeName As XmlQualifiedName

25 [JScript] public function get SchemaTypeName() : XmlQualifiedName;public

function set SchemaTypeName(XmlQualifiedName);

*Description*

Name of the simple type defined in this schema (or another schema indicated by the specified namespace).

The type must be a QName. The type can include a namespace prefix.

SourceUri

UnhandledAttributes

Use

ToString

*Description*

Indicates how the attribute is used.

If specified, this attribute must have one of the following values: Enum

Description **None** Attribute has no value.

XmlSchemaAttributeGroup class (System.Xml.Schema)

ToString

*Description*

Class for attribute groups. Groups a set of attribute declarations so that they can be incorporated as a group into complex type definitions. Represents the W3C **attributeGroup** element.

An attribute group can be defined only as a child of the **schema** element. In this case, the **name** attribute must be present and contain the **attribute** , **attributeGroup** , or **anyAttribute** elements that make up the attribute group.

XmlSchemaAttributeGroup

*Example Syntax:*

ToString

[C#] public XmlSchemaAttributeGroup();

[C++] public: XmlSchemaAttributeGroup();

[VB] Public Sub New()

[JScript] public function XmlSchemaAttributeGroup();

Annotation

AnyAttribute

ToString

*Description*

Allows an **XmlSchemaAnyAttribute** class to be used interchangeably for the attribute.

Attributes

ToString

[C#] public XmlSchemaObjectCollection Attributes {get;}

[C++] public: \_\_property XmlSchemaObjectCollection\* get\_Attributes();

[VB] Public ReadOnly Property Attributes As XmlSchemaObjectCollection

[JScript] public function get Attributes() : XmlSchemaObjectCollection;

### *Description*

Collection of attributes for the attribute group. Contains

**XmlSchemaAttribute** and **XmlSchemaAttributeGroupRef** elements.

Id

LineNumber

LinePosition

Name

ToString

### *Description*

The name of the attribute group.

The name must be an NCName as defined in the XML Namespaces specification. When a **complexType** or **attributeGroup** element includes an attribute group, the name of the **attributeGroup** element is used for the **ref** attribute. This value must be a NCName.

SourceUri

UnhandledAttributes

XmlSchemaAttributeGroupRef class (System.Xml.Schema)

ToString

### *Description*

Class for attribute group reference. Represents the W3C

**AttributeGroupRef** element.

**XmlSchemaAttributeGroupRef** must be present if the attribute group includes an **attributeGroup** or **complexType** element.

XmlSchemaAttributeGroupRef

*Example Syntax:*

ToString

[C#] public XmlSchemaAttributeGroupRef();

[C++] public: XmlSchemaAttributeGroupRef();

[VB] Public Sub New()

[JScript] public function XmlSchemaAttributeGroupRef();

Annotation

Id

LineNumber

LinePosition

RefName

ToString

### *Description*

The referenced name for the **AttributeGroupRef** element.

SourceUri

UnhandledAttributes

XmlSchemaChoice class (System.Xml.Schema)

ToString

*Description*

Allows only one of its children to appear in an instance. Represents the W3C **choice** (compositor) element.

XmlSchemaChoice

*Example Syntax:*

ToString

[C#] public XmlSchemaChoice();

[C++] public: XmlSchemaChoice();

[VB] Public Sub New()

[JScript] public function XmlSchemaChoice();

Annotation

Id

Items

ToString

*Description*

Collection of the elements contained with the compositor (choice):

**XmlSchemaElement** , **XmlSchemaGroupRef** , **XmlSchemaChoice** ,

**XmlSchemaSequence** , or **XmlSchemaAny** .

LineNumber

LinePosition  
 MaxOccurs  
 MaxOccursString  
 MinOccurs  
 MinOccursString  
 SourceUri  
 UnhandledAttributes  
 XmlSchemaCollection class (System.Xml.Schema)  
 ToString

## Description

Contains a cache of XML Schema Definition language (XSD) and XML-Data Reduced Language (XDR) schemas. This class cannot be inherited.

Schemas are loaded using the **System.Xml.Schema.XmlSchemaCollection.Add(System.String,System.String)** method, at which time the schema is associated with a namespace URI. For XSD schemas, this will typically be the schema's **targetNamespace** property.

XmlSchemaCollection

*Example Syntax:*

ToString

[C#] public XmlSchemaCollection();

[C++] public: XmlSchemaCollection();

[VB] Public Sub New()



[JScript] public function XmlSchemaCollection(); Initializes a new instance of the **XmlSchemaCollection** class.

#### *Description*

Initializes a new instance of the **XmlSchemaCollection** class.

**XmlSchemaCollection**

*Example Syntax:*

ToString

[C#] public XmlSchemaCollection(XmlNameTable nametable);

[C++] public: XmlSchemaCollection(XmlNameTable\* nametable);

[VB] Public Sub New(ByVal nametable As XmlNameTable)

[JScript] public function XmlSchemaCollection(nametable : XmlNameTable);

#### *Description*

Initializes a new instance of the **XmlSchemaCollection** class with the specified **System.Xml.XmlNameTable** . The **XmlNameTable** is used when loading schemas. The **XmlNameTable** to use.

Count

ToString

[C#] public int Count {get;}

[C++] public: \_\_property int get\_Count();

[VB] Public ReadOnly Property Count As Integer

[JScript] public function get Count() : int;

1  
2 *Description*

3 Gets the number of namespaces defined in this collection.

4 Item

5 ToString

6  
7 [C#] public XmlSchema this[string ns] {get;}

8 [C++] public: \_\_property XmlSchema\* get\_Item(String\* ns);

9 [VB] Public Default ReadOnly Property Item(ByVal ns As String) As XmlSchema

10 [JScript] returnValue = XmlSchemaCollectionObject.Item(ns);

11  
12 *Description*

13 Gets the **System.Xml.Schema.XmlSchema** associated with the given  
14 namespace URI. The namespace URI associated with the schema you want to  
15 return. This will typically be the targetNamespace of the schema.

16 NameTable

17 ToString

18  
19 [C#] public XmlNameTable NameTable {get;}

20 [C++] public: \_\_property XmlNameTable\* get\_NameTable();

21 [VB] Public ReadOnly Property NameTable As XmlNameTable

22 [JScript] public function get NameTable() : XmlNameTable;

23  
24 *Description*

Gets the default **XmlNameTable** used by the **XmlSchemaCollection** when loading new schemas.

**ToString**

[C#] public event ValidationEventHandler ValidationEventHandler;

[C++] public: \_\_event ValidationEventHandler\* ValidationEventHandler;

[VB] Public Event ValidationEventHandler As ValidationEventHandler

### *Description*

Sets an event handler for receiving information about the XML-Data Reduced Language (XDR) and XML Schema Definition language (XSD) schema validation errors.

These events occur when the schemas are added to the collection. If an event handler is not provided, an **System.Xml.XmlException** is thrown on any validation errors where the **System.Xml.Schema.ValidationEventArgs.Severity** is **XmlSeverityType.Error**. To specify an event handler, define a callback function and add it to the **ValidationEventHandler**.

**Add**

[C#] public XmlSchema Add(XmlSchema schema);

[C++] public: XmlSchema\* Add(XmlSchema\* schema);

[VB] Public Function Add(ByVal schema As XmlSchema) As XmlSchema

[JScript] public function Add(schema : XmlSchema) : XmlSchema;

### *Description*

Adds the **System.Xml.Schema.XmlSchema** to the collection.

*Return Value:* The **XmlSchema** object.

The **targetNamespace** attribute is used to identify this schema. The **XmlSchema** to add to the collection.

Add

[C#] public void Add(XmlSchemaCollection schema);

[C++] public: void Add(XmlSchemaCollection\* schema);

[VB] Public Sub Add(ByVal schema As XmlSchemaCollection)

[JScript] public function Add(schema : XmlSchemaCollection);

### *Description*

Adds all the namespaces defined in the given collection (including their associated schemas) to this collection. The **XmlSchemaCollection** you want to add to this collection.

Add

[C#] public XmlSchema Add(string ns, string uri);

[C++] public: XmlSchema\* Add(String\* ns, String\* uri);

[VB] Public Function Add(ByVal ns As String, ByVal uri As String) As

XmlSchema

[JScript] public function Add(ns : String, uri : String) : XmlSchema; Adds the given schema into the schema collection.

### *Description*

1 Adds the schema located by the given URL into the schema collection.

2 *Return Value:* The **System.Xml.Schema.XmlSchema** added to the schema  
3 collection.

4 If *ns* has already been associated with another schema in the collection, the  
5 schema being added replaces the original schema in the collection. For example, in  
6 the following C# code, authors.xsd is removed from the collection and names.xsd  
7 is added. The namespace URI associated with the schema. For XML Schema  
8 Definition language (XSD) schemas, this will typically be the targetNamespace.  
9 The URL that specifies the schema to load.

10 Add

11  
12 [C#] public XmlSchema Add(string ns, XmlReader reader);

13 [C++] public: XmlSchema\* Add(String\* ns, XmlReader\* reader);

14 [VB] Public Function Add(ByVal ns As String, ByVal reader As XmlReader) As  
15 XmlSchema

16 [JScript] public function Add(ns : String, reader : XmlReader) : XmlSchema;

17  
18 *Description*

19 Adds the given schema into the schema collection.

20 *Return Value:* The **System.Xml.Schema.XmlSchema** added to the schema  
21 collection.

22 If *ns* has already been associated with another schema in the collection, the  
23 schema being added replaces the original schema in the collection. The namespace  
24 URI associated with the schema. For XML Schema Definition language (XSD)  
25

schemas, this will typically be the targetNamespace. **System.Xml.XmlReader** containing the schema to add.

#### Contains

[C#] public bool Contains(string ns);

[C++] public: bool Contains(String\* ns);

[VB] Public Function Contains(ByVal ns As String) As Boolean

[JScript] public function Contains(ns : String) : Boolean;

#### *Description*

Gets a value indicating whether the specified schema is in the collection.

*Return Value:* **true** if the schema is in the collection; otherwise, **false**. The namespace URI associated with the schema. For XML Schema Definition language (XSD) schemas, this will typically be the target namespace.

#### Contains

[C#] public bool Contains(XmlSchema schema);

[C++] public: bool Contains(XmlSchema\* schema);

[VB] Public Function Contains(ByVal schema As XmlSchema) As Boolean

[JScript] public function Contains(schema : XmlSchema) : Boolean; Gets a value indicating whether the specified schema is in the collection.

#### *Description*

Gets a value indicating whether the specified

**System.Xml.Schema.XmlSchema** is in the collection.

*Return Value:* **true** if the schema is in the collection; otherwise, **false** . The name of the schema.

### CopyTo

[C#] public void CopyTo(XmlSchema[] array, int index);

[C++] public: void CopyTo(XmlSchema\* array[], int index);

[VB] Public Sub CopyTo(ByVal array() As XmlSchema, ByVal index As Integer)

[JScript] public function CopyTo(array : XmlSchema[], index : int);

### Description

Copies all the **XmlSchema** objects from this collection into the given array starting at the given index. The array to copy the objects to. The index in *array* where copying will begin.

### GetEnumerator

[C#] public XmlSchemaCollectionEnumerator GetEnumerator();

[C++] public: XmlSchemaCollectionEnumerator\* GetEnumerator();

[VB] Public Function GetEnumerator() As XmlSchemaCollectionEnumerator

[JScript] public function GetEnumerator() : XmlSchemaCollectionEnumerator;

### Description

Provides support for the "for each" style iteration over the collection of schemas.

*Return Value:* An enumerator for iterating over all schemas in the current collection.

## ICollection.CopyTo

[C#] void ICollection.CopyTo(Array array, int index);

[C++] void ICollection::CopyTo(Array\* array, int index);

[VB] Sub CopyTo(ByVal array As Array, ByVal index As Integer) Implements

ICollection.CopyTo

[JScript] function ICollection.CopyTo(array : Array, index : int);

## IEnumerable.GetEnumerator

[C#] IEnumerator IEnumerable.GetEnumerator();

[C++] IEnumerator\* IEnumerable::GetEnumerator();

[VB] Function GetEnumerator() As IEnumerator Implements

IEnumerable.GetEnumerator

[JScript] function IEnumerable.GetEnumerator() : IEnumerator;

XmlSchemaCollectionEnumerator class (System.Xml.Schema)

ToString

## Description

Supports a simple iteration over a collection. This class cannot be inherited.

Current

ToString

[C#] public XmlSchema Current {get;}

[C++] public: \_\_property XmlSchema\* get\_Current();



1 [VB] Public ReadOnly Property Current As XmlSchema

2 [JScript] public function get Current() : XmlSchema;

3  
4 *Description*

5 Gets the current **System.Xml.Schema.XmlSchema** in the collection.

6 MoveNext

7  
8 [C#] public bool MoveNext();

9 [C++] public: bool MoveNext();

10 [VB] Public Function MoveNext() As Boolean

11 [JScript] public function MoveNext() : Boolean;

12  
13 *Description*

14 Advances the enumerator to the next schema in the collection.

15 *Return Value:* **true** if the move was successful; **false** if the enumerator has passed  
16 the end of the collection.

17 IEnumerator.MoveNext

18  
19 [C#] bool IEnumerator.MoveNext();

20 [C++] bool IEnumerator::MoveNext();

21 [VB] Function MoveNext() As Boolean Implements IEnumerator.MoveNext

22 [JScript] function IEnumerator.MoveNext() : Boolean;

23 IEnumerator.Reset

24  
25 [C#] void IEnumerator.Reset();

1 [C++] void IEnumerator::Reset();  
2 [VB] Sub Reset() Implements IEnumerator.Reset  
3 [JScript] function IEnumerator.Reset();  
4 XmlSchemaComplexContent class (System.Xml.Schema)  
5 ToString

6  
7  
8 *Description*

9 Class that represents complex content model for complex types. Contains  
10 extensions or restrictions on a complex type that has mixed content or elements  
11 only.

12 XmlSchemaComplexContent

13 *Example Syntax:*

14 ToString

15  
16 [C#] public XmlSchemaComplexContent();  
17 [C++] public: XmlSchemaComplexContent();  
18 [VB] Public Sub New()  
19 [JScript] public function XmlSchemaComplexContent();  
20 Annotation  
21 Content  
22 ToString

23  
24  
25 *Description*

One of either the **XmlSchemaComplexContentRestriction** or **XmlSchemaComplexContentExtension** classes.

Id

IsMixed

ToString

#### *Description*

Indicates that this type has a mixed content model. Character data is allowed to appear between the child elements of the complex type.

This **mixed** attribute can override the **mixed** attribute value on the containing **complexType** element. The default is **false** .

LineNumber

LinePosition

SourceUri

UnhandledAttributes

XmlSchemaComplexContentExtension class (System.Xml.Schema)

ToString

#### *Description*

Class for complex types with a simple content model that is extended by extension.

XmlSchemaComplexContentExtension

*Example Syntax:*

ToString

[C#] public XmlSchemaComplexContentExtension();

[C++] public: XmlSchemaComplexContentExtension();

[VB] Public Sub New()

[JScript] public function XmlSchemaComplexContentExtension();

Annotation

AnyAttribute

ToString

### Description

Allows an **XmlSchemaAnyAttribute** class to be used for the attribute value.

Attributes

ToString

[C#] public XmlSchemaObjectCollection Attributes {get;}

[C++] public: \_\_property XmlSchemaObjectCollection\* get\_Attributes();

[VB] Public ReadOnly Property Attributes As XmlSchemaObjectCollection

[JScript] public function get Attributes() : XmlSchemaObjectCollection;

### Description

Contains **XmlSchemaAttribute** and **XmlSchemaAttributeGroupRef** classes. Collection of attributes for the **complexType** element.

BaseTypeName

ToString

[C#] public XmlQualifiedName BaseTypeName {get; set;}

[C++] public: \_\_property XmlQualifiedName\* get\_BaseTypeName();public:

\_\_property void set\_BaseTypeName(XmlQualifiedName\*);

[VB] Public Property BaseTypeName As XmlQualifiedName

[JScript] public function get BaseTypeName() : XmlQualifiedName;public

function set BaseTypeName(XmlQualifiedName);

### *Description*

Name of a built-in data type, simple type, or complex type. This **complexType** is derived from the type specified by the base value.

The base value must be a QName.

Id

LineNumber

LinePosition

Particle

ToString

### *Description*

One of the **XmlSchemaGroupRef** , **XmlSchemaChoice** , **XmlSchemaAll** , or **XmlSchemaSequence** classes.

SourceUri

UnhandledAttributes

XmlSchemaComplexContentRestriction class (System.Xml.Schema)

ToString

### *Description*

Class for complex types with complex content model that are extended by restriction. Restricts the contents of the **complexType** element to a subset of the inherited **complexType** element.

XmlSchemaComplexContentRestriction

### *Example Syntax:*

ToString

[C#] public XmlSchemaComplexContentRestriction();

[C++] public: XmlSchemaComplexContentRestriction();

[VB] Public Sub New()

[JScript] public function XmlSchemaComplexContentRestriction();

Annotation

AnyAttribute

ToString

### *Description*

Allows an **XmlSchemaAnyAttribute** to be used for the attribute.

Attributes

ToString

[C#] public XmlSchemaObjectCollection Attributes {get;}

[C++] public: \_\_property XmlSchemaObjectCollection\* get\_Attributes();

[VB] Public ReadOnly Property Attributes As XmlSchemaObjectCollection

[JScript] public function get Attributes() : XmlSchemaObjectCollection;

### *Description*

Contains **XmlSchemaAttribute** and **XmlSchemaAttributeGroupRef** classes. Collection of attributes for the complex type.

BaseTypeName

ToString

[C#] public XmlQualifiedName BaseTypeName {get; set;}

[C++] public: \_\_property XmlQualifiedName\* get\_BaseTypeName();public:

\_\_property void set\_BaseTypeName(XmlQualifiedName\*);

[VB] Public Property BaseTypeName As XmlQualifiedName

[JScript] public function get BaseTypeName() : XmlQualifiedName;public

function set BaseTypeName(XmlQualifiedName);

### *Description*

Name of a built-in data type, simple type, or complex type. The complex type is derived from the type specified by the base value.

The base value must be a QName.

Id

LineNumber

LinePosition

Particle

ToString

### Description

One of the **XmlSchemaGroupRef** , **XmlSchemaChoice** , **XmlSchemaAll** , or **XmlSchemaSequence** classes.

SourceUri

UnhandledAttributes

XmlSchemaComplexType class (System.Xml.Schema)

ToString

### Description

Class for complex types. Defines a complex type that determines the set of attributes and content of an element.

An element can reference a simple type in the **type** attribute. A complex type is essentially a type definition for elements. An element can be declared with a **type** attribute that refers to a **complexType** element that defines the structure, content, and attributes of that element. An element can also take a reference to a **simpleType** element in its type attribute.

XmlSchemaComplexType

*Example Syntax:*



ToString

[C#] public XmlSchemaComplexType();

[C++] public: XmlSchemaComplexType();

[VB] Public Sub New()

[JScript] public function XmlSchemaComplexType();

*Description*

Used to create a **complexType** element.

Annotation

AnyAttribute

ToString

*Description*

Complex type can contain any attributes from the specified namespace(s).

Attributes

ToString

[C#] public XmlSchemaObjectCollection Attributes {get;}

[C++] public: \_\_property XmlSchemaObjectCollection\* get\_Attributes();

[VB] Public ReadOnly Property Attributes As XmlSchemaObjectCollection

[JScript] public function get Attributes() : XmlSchemaObjectCollection;

*Description*

Collection of attributes for the complex type. Contains  
**XmlSchemaAttribute** and **XmlSchemaAttributeGroupRef** classes.

If the **group** , **sequence** , **choice** , or **all** element is specified as the child element, these are the attributes for the complex type.

AttributeUses

ToString

[C#] public XmlSchemaObjectTable AttributeUses {get;}

[C++] public: \_\_property XmlSchemaObjectTable\* get\_AttributeUses();

[VB] Public ReadOnly Property AttributeUses As XmlSchemaObjectTable

[JScript] public function get AttributeUses() : XmlSchemaObjectTable;

### *Description*

Uses all the attributes from this complex type and its base derived types.

For example, a combination of all the attributes from the **Attributes** property from each type.

AttributeWildcard

ToString

[C#] public XmlSchemaAnyAttribute AttributeWildcard {get;}

[C++] public: \_\_property XmlSchemaAnyAttribute\* get\_AttributeWildcard();

[VB] Public ReadOnly Property AttributeWildcard As XmlSchemaAnyAttribute

[JScript] public function get AttributeWildcard() : XmlSchemaAnyAttribute;

### *Description*

The **anyAttribute** element from this complex type and its base derived types.

For example, **anyAttribute** from the base **anyAttribute** property.

BaseSchemaType

Block

ToString

### *Description*

Indicates the type of derivation. The **block** attribute prevents a complex type that has the specified type of derivation from being used in place of this complex type.

The value can contain **all** or a list that is a subset of extension/restriction.

BlockResolved

ToString

```
[C#] public XmlSchemaDerivationMethod BlockResolved {get;}
```

```
[C++] public: __property XmlSchemaDerivationMethod get_BlockResolved();
```

```
[VB] Public ReadOnly Property BlockResolved As XmlSchemaDerivationMethod
```

```
[JScript] public function get BlockResolved() : XmlSchemaDerivationMethod;
```

### *Description*

The value after an element has been compiled to post-schema info set. This value indicates how the type is enforced when **xsi:type** is used in the instance document.

This value is either from the type itself or, if the type is not defined, the type is taken from the **schema** element. For example, if the value is restricted, only the specifically defined type can be used, not types derived from the specifically defined type.

ContentModel

ToString

[C#] public XmlSchemaContentModel ContentModel {get; set;}

[C++] public: \_\_property XmlSchemaContentModel\* get\_ContentModel();public:

\_\_property void set\_ContentModel(XmlSchemaContentModel\*);

[VB] Public Property ContentModel As XmlSchemaContentModel

[JScript] public function get ContentModel() : XmlSchemaContentModel;public

function set ContentModel(XmlSchemaContentModel);

### Description

Indicates the content model type. One of the **XmlSchemaSimpleContent** or **XmlSchemaComplexContent** classes.

This is mutually exclusive with the **System.Xml.Schema.Particle** property.

ContentType

ToString

[C#] public XmlSchemaContentType ContentType {get;}

[C++] public: \_\_property XmlSchemaContentType get\_ContentType();

[VB] Public ReadOnly Property ContentType As XmlSchemaContentType

[JScript] public function get ContentType() : XmlSchemaContentType;

### Description

The content model of the complex type.

This is the content in the post-schema Infoset.

ContentTypeParticle

ToString

[C#] public XmlSchemaParticle ContentTypeParticle {get;}

[C++] public: \_\_property XmlSchemaParticle\* get\_ContentTypeParticle();

[VB] Public ReadOnly Property ContentTypeParticle As XmlSchemaParticle

[JScript] public function get ContentTypeParticle() : XmlSchemaParticle;

### Description

The particle that is determined after the content models of the most derived types and the base complex type have been resolved according to the rules defined in the W3C XML Schema Definition language (XSD) specification.

Datatype

DerivedBy

Final

FinalResolved

Id

IsAbstract

ToString

*Description*

Indicates if the **complexType** element can be used in the instance document.

If **true** , an element cannot use this **complexType** element directly and must use a complex type that is derived from this **complexType** element. The default is **false** .

IsMixed

ToString

[C#] public override bool IsMixed {get; set;}

[C++] public: \_\_property virtual bool get\_IsMixed();public: \_\_property virtual void set\_IsMixed(bool);

[VB] Overrides Public Property IsMixed As Boolean

[JScript] public function get IsMixed() : Boolean;public function set IsMixed(Boolean);

*Description*

Indicates that this type has a mixed content model (markup within the content).

Indicates if character data can appear between the child elements of this **complexType** . If the **simpleContent** element is a child element, the **mixed** attribute is not allowed ( **false** ). If the **complexContent** element is a child

element, the **mixed** attribute can be overridden by the **mixed** attribute on the **complexContent** element.

- LineNumber
- LinePosition
- Name
- Particle
- ToString

*Description*

Indicates the compositor type as one of the **XmlSchemaGroupRef** , **XmlSchemaChoice** , **XmlSchemaAll** , or **XmlSchemaSequence** classes.

Particles must be one of the following compositors: Element Description  
**group** The complex type contains the elements defined in the referenced group.

- QualifiedName
- SourceUri
- UnhandledAttributes
- XmlSchemaContent class (System.Xml.Schema)
- ToString

*Description*

An abstract class for schema content.

XmlSchemaContent

*Example Syntax:*

ToString

[C#] protected XmlSchemaContent();

[C++] protected: XmlSchemaContent();

[VB] Protected Sub New()

[JScript] protected function XmlSchemaContent();

Annotation

Id

LineNumber

LinePosition

SourceUri

UnhandledAttributes

XmlSchemaContentModel class (System.Xml.Schema)

ToString

### Description

An abstract class for the schema content model.

XmlSchemaContentModel

*Example Syntax:*

ToString

[C#] protected XmlSchemaContentModel();

[C++] protected: XmlSchemaContentModel();



[VB] Protected Sub New()

[JScript] protected function XmlSchemaContentModel();

Annotation

Content

ToString

*Description*

Provides the content of the type.

Id

LineNumber

LinePosition

SourceUri

UnhandledAttributes

XmlSchemaContentProcessing enumeration (System.Xml.Schema)

ToString

*Description*

Provides information about the constraints placed on the replacement elements.

ToString

[C#] public const XmlSchemaContentProcessing Lax;

[C++] public: const XmlSchemaContentProcessing Lax;

[VB] Public Const Lax As XmlSchemaContentProcessing

[JScript] public var Lax : XmlSchemaContentProcessing;

#### *Description*

Lax. The document should recognize element types but can ignore element types that it does not recognize.

ToString

[C#] public const XmlSchemaContentProcessing None;

[C++] public: const XmlSchemaContentProcessing None;

[VB] Public Const None As XmlSchemaContentProcessing

[JScript] public var None : XmlSchemaContentProcessing;

#### *Description*

ToString

[C#] public const XmlSchemaContentProcessing Skip;

[C++] public: const XmlSchemaContentProcessing Skip;

[VB] Public Const Skip As XmlSchemaContentProcessing

[JScript] public var Skip : XmlSchemaContentProcessing;

#### *Description*

Skip. The document must consist of well-formed XML and is not validated by the schema.

ToString

```
[C#] public const XmlSchemaContentProcessing Strict;
[C++] public: const XmlSchemaContentProcessing Strict;
[VB] Public Const Strict As XmlSchemaContentProcessing
[JScript] public var Strict : XmlSchemaContentProcessing;
```

*Description*

Strict. The document must be schema-valid based on the schema definition derived from the namespace-qualified element name.

XmlSchemaContentType enumeration (System.Xml.Schema)

ToString

*Description*

Enumerations for the content model of the complex type. This represents the content in the post-schema infoset.

If the **IsMixed** property is set to **True** on the **ContentModel** property that has the type **XmlSchemaComplexContent** , the value returned here is **Mixed** . If the **IsMixed** property is **False** , the value is either **Empty** or **ElementOnly** .

ToString

```
[C#] public const XmlSchemaContentType ElementOnly;
[C++] public: const XmlSchemaContentType ElementOnly;
[VB] Public Const ElementOnly As XmlSchemaContentType
```

1 [JScript] public var ElementOnly : XmlSchemaContentType;

2  
3 *Description*

4 Element only content.

5 ToString

6  
7 [C#] public const XmlSchemaContentType Empty;

8 [C++] public: const XmlSchemaContentType Empty;

9 [VB] Public Const Empty As XmlSchemaContentType

10 [JScript] public var Empty : XmlSchemaContentType;

11  
12 *Description*

13 Empty content.

14 ToString

15  
16 [C#] public const XmlSchemaContentType Mixed;

17 [C++] public: const XmlSchemaContentType Mixed;

18 [VB] Public Const Mixed As XmlSchemaContentType

19 [JScript] public var Mixed : XmlSchemaContentType;

20  
21 *Description*

22 Mixed content.

23 ToString

24  
25 [C#] public const XmlSchemaContentType TextOnly;

```

1 [C++] public: const XmlSchemaContentType TextOnly;
2 [VB] Public Const TextOnly As XmlSchemaContentType
3 [JScript] public var TextOnly : XmlSchemaContentType;
4

```

#### *Description*

Text only content.

XmlSchemaDatatype class (System.Xml.Schema)

ToString

#### *Description*

Represents the abstract class for mapping XML Schema Definition language (XSD) and common language runtime types.

XmlSchemaDatatype

*Example Syntax:*

ToString

```

18 [C#] protected XmlSchemaDatatype();
19 [C++] protected: XmlSchemaDatatype();
20 [VB] Protected Sub New()
21 [JScript] protected function XmlSchemaDatatype();
22
23     TokenizedType
24
25     ToString
26
27 [C#] public abstract XmlTokenizedType TokenizedType {get;}

```

1 [C++] public: \_\_property virtual XmlTokenizedType get\_TokenizedType() = 0;

2 [VB] MustOverride Public ReadOnly Property TokenizedType As

3 XmlTokenizedType

4 [JScript] public abstract function get TokenizedType() : XmlTokenizedType;

5  
6 *Description*

7 Represents the type for the string as specified in the XML 1.0 specification.

8 This allows the string to be read as a particular XML type, for example, a  
9 CDATA section type.

10 ValueType

11 ToString

12  
13 [C#] public abstract Type ValueType {get;}

14 [C++] public: \_\_property virtual Type\* get\_ValueType() = 0;

15 [VB] MustOverride Public ReadOnly Property ValueType As Type

16 [JScript] public abstract function get ValueType() : Type;

17  
18 *Description*

19 Returns the common language runtime type for the specified XML Schema  
20 Definition language (XSD) type.

21 The table maps the XSD and common language runtime types.

22 ParseValue

23  
24 [C#] public abstract object ParseValue(string s, XmlNameTable nameTable,

25 XmlNamespaceManager nsmgr);

[C++] public: virtual Object\* ParseValue(String\* s, XmlNameTable\* nameTable, XmlNamespaceManager\* nsmgr) = 0;

[VB] MustOverride Public Function ParseValue(ByVal s As String, ByVal nameTable As XmlNameTable, ByVal nsmgr As XmlNamespaceManager) As Object

[JScript] public abstract function ParseValue(s : String, nameTable : XmlNameTable, nsmgr : XmlNamespaceManager) : Object;

#### *Description*

Parses values from the **XmlNameTable** . string

#### **XmlNameTableXmlNamespaceManager**

XmlSchemaDerivationMethod enumeration (System.Xml.Schema)

ToString

#### *Description*

Prevents derivations by restriction from replacing the complex type in an instance.

**XmlSchemaBlock** does not prevent the replacement of complex types that are derived by extension.

ToString

[C#] public const XmlSchemaDerivationMethod All;

[C++] public: const XmlSchemaDerivationMethod All;

[VB] Public Const All As XmlSchemaDerivationMethod

[JScript] public var All : XmlSchemaDerivationMethod;

*Description*

No restrictions on derivations from this type.

ToString

[C#] public const XmlSchemaDerivationMethod Empty;

[C++] public: const XmlSchemaDerivationMethod Empty;

[VB] Public Const Empty As XmlSchemaDerivationMethod

[JScript] public var Empty : XmlSchemaDerivationMethod;

*Description*

Sets a derived setting to an empty string that allows any extension or restriction regardless of the **schema** element's default setting.

ToString

[C#] public const XmlSchemaDerivationMethod Extension;

[C++] public: const XmlSchemaDerivationMethod Extension;

[VB] Public Const Extension As XmlSchemaDerivationMethod

[JScript] public var Extension : XmlSchemaDerivationMethod;

*Description*

Can only be derived by Extension.

ToString



```
[C#] public const XmlSchemaDerivationMethod List;
[C++] public: const XmlSchemaDerivationMethod List;
[VB] Public Const List As XmlSchemaDerivationMethod
[JScript] public var List : XmlSchemaDerivationMethod;
```

*Description*

Can only be derived by List.

ToString

```
[C#] public const XmlSchemaDerivationMethod None;
[C++] public: const XmlSchemaDerivationMethod None;
[VB] Public Const None As XmlSchemaDerivationMethod
[JScript] public var None : XmlSchemaDerivationMethod;
```

*Description*

The normal default setting that can be overridden by the **schema** element's default setting.

ToString

```
[C#] public const XmlSchemaDerivationMethod Restriction;
[C++] public: const XmlSchemaDerivationMethod Restriction;
[VB] Public Const Restriction As XmlSchemaDerivationMethod
[JScript] public var Restriction : XmlSchemaDerivationMethod;
```

*Description*

Can only be derived by Restriction.

ToString

```
[C#] public const XmlSchemaDerivationMethod Substitution;
[C++] public: const XmlSchemaDerivationMethod Substitution;
[VB] Public Const Substitution As XmlSchemaDerivationMethod
[JScript] public var Substitution : XmlSchemaDerivationMethod;
```

*Description*

Can only be derived by Substitution.

ToString

```
[C#] public const XmlSchemaDerivationMethod Union;
[C++] public: const XmlSchemaDerivationMethod Union;
[VB] Public Const Union As XmlSchemaDerivationMethod
[JScript] public var Union : XmlSchemaDerivationMethod;
```

*Description*

Can only be derived by Union.

XmlSchemaDocumentation class (System.Xml.Schema)

ToString

### *Description*

Class that specifies information to be read or used by humans within an annotation. Represents the W3C **documentation** element.

Information supplied within the **XmlSchemaDocumentation** class is not used in validation. However, it provides a standardized way to supply information that can be retrieved by tools.

XmlSchemaDocumentation

#### *Example Syntax:*

ToString

[C#] public XmlSchemaDocumentation();

[C++] public: XmlSchemaDocumentation();

[VB] Public Sub New()

[JScript] public function XmlSchemaDocumentation();

Language

ToString

[C#] public string Language {get; set;}

[C++] public: \_\_property String\* get\_Language();public: \_\_property void set\_Language(String\*);

[VB] Public Property Language As String

[JScript] public function get Language() : String;public function set Language(String);

*Description*

Provides the **xml:lang** attribute.

LineNumber

LinePosition

Markup

ToString

*Description*

Returns an array of **XmlNodes** that represents the document text markup.

Source

ToString

[C#] public string Source {get; set;}

[C++] public: \_\_property String\* get\_Source();public: \_\_property void  
set\_Source(String\*);

[VB] Public Property Source As String

[JScript] public function get Source() : String;public function set Source(String);

*Description*

Provides the URI source of the application information.

Source must be a URI reference.

SourceUri

XmlSchemaElement class (System.Xml.Schema)

ToString

*Description*

Class for elements.

XmlSchemaElement

*Example Syntax:*

ToString

[C#] public XmlSchemaElement();

[C++] public: XmlSchemaElement();

[VB] Public Sub New()

[JScript] public function XmlSchemaElement();

Annotation

Block

ToString

*Description*

Attribute used to block a type derivation.

This attribute can contain the following values: Enum Description **None**

The normal default block setting can be overridden by the schema element  
blockDefault setting.

BlockResolved

ToString

```
[C#] public XmlSchemaDerivationMethod BlockResolved {get;}
[C++] public: __property XmlSchemaDerivationMethod get_BlockResolved();
[VB] Public ReadOnly Property BlockResolved As XmlSchemaDerivationMethod
[JavaScript] public function get BlockResolved() : XmlSchemaDerivationMethod;
```

### Description

The value after an element has been compiled to post-schema info set. This value is either from the type itself or, if not defined on the type, taken from the **schema** element.

This value indicates how the type is enforced when **xsi:type** is used in the instance document. For example, if the value is restricted, then only the specific defined type can be used, not types derived from the specific defined type.

### Constraints

### ToString

```
[C#] public XmlSchemaObjectCollection Constraints {get;}
[C++] public: __property XmlSchemaObjectCollection* get_Constraints();
[VB] Public ReadOnly Property Constraints As XmlSchemaObjectCollection
[JavaScript] public function get Constraints() : XmlSchemaObjectCollection;
```

### Description

Returns a collection of constraints on the element.

Returns a collection of constraints on the element. Can be any of **XmlSchemaKey** , **XmlSchemaKeyref** , or **XmlSchemaUnique** .

DefaultValue

ToString

[C#] public string DefaultValue {get; set;}

[C++] public: \_\_property String\* get\_DefaultValue();public: \_\_property void  
set\_DefaultValue(String\*);

[VB] Public Property DefaultValue As String

[JScript] public function get DefaultValue() : String;public function set  
DefaultValue(String);

### Description

Provides the default value of the element if its content is a simple type or  
the element's content is **textOnly** .

The **fixed** and **default** attributes are mutually exclusive. If the element  
contains a simple type, this value must be a valid value of that type.

ElementType

ToString

[C#] public object ElementType {get;}

[C++] public: \_\_property Object\* get\_ElementType();

[VB] Public ReadOnly Property ElementType As Object

[JScript] public function get ElementType() : Object;

### Description

Returns the correct common runtime library object based upon the  
**SchemaType** for the element.

Final

ToString

```
[C#] public XmlSchemaDerivationMethod Final {get; set;}
```

```
[C++] public: __property XmlSchemaDerivationMethod get_Final();public:
```

```
__property void set_Final(XmlSchemaDerivationMethod);
```

```
[VB] Public Property Final As XmlSchemaDerivationMethod
```

```
[JScript] public function get Final() : XmlSchemaDerivationMethod;public
```

```
function set Final(XmlSchemaDerivationMethod);
```

### *Description*

Indicates that no further derivations are allowed.

This attribute can contain the following values: Enum Description **None**

The normal default final setting can be overridden by the **schema** element  
finalDefault setting.

FinalResolved

ToString

```
[C#] public XmlSchemaDerivationMethod FinalResolved {get;}
```

```
[C++] public: __property XmlSchemaDerivationMethod get_FinalResolved();
```

```
[VB] Public ReadOnly Property FinalResolved As XmlSchemaDerivationMethod
```

```
[JScript] public function get FinalResolved() : XmlSchemaDerivationMethod;
```



## Description

The value after an element has been compiled to post-schema info set. This value is either from the type itself or, if not defined on the type, taken from the **schema** element.

This value indicates how the type is restricted when the type is extended through restriction. If the type is marked as **final**, then this type cannot be extended.

FixedValue

ToString

```
[C#] public string FixedValue {get; set;}
```

```
[C++] public: __property String* get_FixedValue();public: __property void  
set_FixedValue(String*);
```

```
[VB] Public Property FixedValue As String
```

```
[JScript] public function get FixedValue() : String;public function set  
FixedValue(String);
```

## Description

Provides the fixed value.

If the content is a simple type or its content is **textOnly**, then the value is predetermined and cannot change its value. The **fixed** and **default** attributes are mutually exclusive.

Form

ToString

1  
2 [C#] public XmlSchemaForm Form {get; set;}

3 [C++] public: \_\_property XmlSchemaForm get\_Form();public: \_\_property void  
4 set\_Form(XmlSchemaForm);

5 [VB] Public Property Form As XmlSchemaForm

6 [JScript] public function get Form() : XmlSchemaForm;public function set  
7 Form(XmlSchemaForm);

8  
9 *Description*

10 Form for the element.

11 The default value is the value of the **elementFormDefault** attribute for the  
12 **schema** element containing the attribute. The default is **Unqualified** .

13 Id

14 IsAbstract

15 ToString

16  
17  
18 *Description*

19 Indicates abstract type. Indicates whether the element can be used in an  
20 instance document.

21 If this value is **true** , the element cannot appear in the instance document.  
22 The default is **false** .

23 IsNillable

24 ToString

[C#] public bool IsNillable {get; set;}

[C++] public: \_\_property bool get\_IsNillable();public: \_\_property void

set\_IsNillable(bool);

[VB] Public Property IsNillable As Boolean

[JScript] public function get IsNillable() : Boolean;public function set

IsNillable(Boolean);

### *Description*

Indicates that **xsi:null** can occur in the instance data. Indicates if an explicit null value can be assigned to the element.

This applies to element content and not the attributes of the element. The default is **false** .

LineNumber

LinePosition

MaxOccurs

MaxOccursString

MinOccurs

MinOccursString

Name

ToString

### *Description*

Provides the name of the element.

The name must be an NCName as defined in the XML Namespaces specification.

QualifiedName

ToString

[C#] public XmlQualifiedName QualifiedName {get;}

[C++] public: \_\_property XmlQualifiedName\* get\_QualifiedName();

[VB] Public ReadOnly Property QualifiedName As XmlQualifiedName

[JScript] public function get QualifiedName() : XmlQualifiedName;

#### *Description*

Returns the qualified name of the element. Name of the a **simpleType** or **complexType** element defined in this schema (or another schema indicated by the specified namespace).

RefName

ToString

[C#] public XmlQualifiedName RefName {get; set;}

[C++] public: \_\_property XmlQualifiedName\* get\_RefName();public: \_\_property void set\_RefName(XmlQualifiedName\*);

[VB] Public Property RefName As XmlQualifiedName

[JScript] public function get RefName() : XmlQualifiedName;public function set RefName(XmlQualifiedName);

#### *Description*

Provides the reference name of an element declared in this schema (or another schema indicated by the specified namespace).

The **RefName** must be a QName. The **RefName** can include a namespace prefix. The **type** and **refName** attributes are mutually exclusive. To declare an element using an existing element definition, use the **RefName** attribute to specify the existing element definition.

SchemaType

ToString

[C#] public XmlSchemaType SchemaType {get; set;}

[C++] public: \_\_property XmlSchemaType\* get\_SchemaType();public:

\_\_property void set\_SchemaType(XmlSchemaType\*);

[VB] Public Property SchemaType As XmlSchemaType

[JScript] public function get SchemaType() : XmlSchemaType;public function set SchemaType(XmlSchemaType);

### *Description*

Returns the type of the element. This can either be a complex type or a simple type.

The **SchemaType** and **RefName** attributes are mutually exclusive. To declare an element using an existing **simpleType** or **complexType** definition, use the **type** attribute to specify the existing type.

SchemaTypeName

ToString

```

1
2 [C#] public XmlQualifiedName SchemaTypeName {get; set;}
3 [C++] public: __property XmlQualifiedName* get_SchemaTypeName();public:
4 __property void set_SchemaTypeName(XmlQualifiedName*);
5 [VB] Public Property SchemaTypeName As XmlQualifiedName
6 [JScript] public function get SchemaTypeName() : XmlQualifiedName;public
7 function set SchemaTypeName(XmlQualifiedName);
8

```

### *Description*

Name of a built-in data type defined in this schema or another schema indicated by the specified namespace.

SourceUri

SubstitutionGroup

ToString

### *Description*

Name of an element that can be a substitute for this element.

This element must have the same type or a type derived from the type of the specified element. This value must be a QName.

UnhandledAttributes

XmlSchemaEnumerationFacet class (System.Xml.Schema)

ToString

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25

*Description*

Class for defining **enumeration** facets.

XmlSchemaEnumerationFacet

*Example Syntax:*

ToString

[C#] public XmlSchemaEnumerationFacet();

[C++] public: XmlSchemaEnumerationFacet();

[VB] Public Sub New()

[JScript] public function XmlSchemaEnumerationFacet();

Annotation

Id

IsFixed

LineNumber

LinePosition

SourceUri

UnhandledAttributes

Value

XmlSchemaException class (System.Xml.Schema)

ToString

*Description*

Returns detailed information about the schema exception.

XmlSchemaException

*Example Syntax:*

ToString

[C#] protected XmlSchemaException(SerializationInfo info, StreamingContext context);

[C++] protected: XmlSchemaException(SerializationInfo\* info, StreamingContext context);

[VB] Protected Sub New(ByVal info As SerializationInfo, ByVal context As StreamingContext)

[JScript] protected function XmlSchemaException(info : SerializationInfo, context : StreamingContext); Streams all the **XmlSchemaException** properties into the **SerializationInfo** class for the given **StreamingContext** .

*Description*

Constructs a new **XmlSchemaException** object with the given **SerializationInfo** and **StreamingContext** information that contains all the properties of the **XmlSchemaException** .

XmlSchemaException

*Example Syntax:*

ToString

[C#] public XmlSchemaException(string message, Exception innerException);

[C++] public: XmlSchemaException(String\* message, Exception\*



1 innerException);

2 [VB] Public Sub New(ByVal message As String, ByVal innerException As

3 Exception)

4 [JScript] public function XmlSchemaException(message : String, innerException :

5 Exception);

6  
7 *Description*

8     The formatted error message describing the error code.

9     HelpLink

10    HRESULT

11    InnerException

12    LineNumber

13    ToString

14  
15  
16 *Description*

17     Gets the line number indicating where the error occurred.

18     LinePosition

19     ToString

20  
21 [C#] public int LinePosition {get;}

22 [C++] public: \_\_property int get\_LinePosition();

23 [VB] Public ReadOnly Property LinePosition As Integer

24 [JScript] public function get LinePosition() : int;

25

*Description*

Gets the line position indicating where the error occurred.

Message

ToString

[C#] public override string Message {get;}

[C++] public: \_\_property virtual String\* get\_Message();

[VB] Overrides Public ReadOnly Property Message As String

[JScript] public function get Message() : String;

*Description*

Gets the formatted error message describing the

**System.Xml.Schema.XmlSchemaException.ErrorCode** .

Source

SourceSchemaObject

ToString

*Description*

The **XmlSchemaObject** that produced the **XmlSchemaException** .

This can be queried for the **LineNumber** , **LinePosition** , and **SourceUri** of the XML Schema Definition language (XSD) schema file. A return value of **null** indicates a syntactic parsing validation error in the Schema Object Model (SOM).

SourceUri

ToString

[C#] public string SourceUri {get;}

[C++] public: \_\_property String\* get\_SourceUri();

[VB] Public ReadOnly Property SourceUri As String

[JScript] public function get SourceUri() : String;

### Description

The location of the file used to load the schema.

StackTrace

TargetSite

GetObjectData

[C#] public override void GetObjectData(SerializationInfo info, StreamingContext context);

[C++] public: void GetObjectData(SerializationInfo\* info, StreamingContext context);

[VB] Overrides Public Sub GetObjectData(ByVal info As SerializationInfo, ByVal context As StreamingContext)

[JScript] public override function GetObjectData(info : SerializationInfo, context : StreamingContext);

### Description

Streams all the **XmlSchemaException** properties into the **SerializationInfo** class for the given **StreamingContext** .

XmlSchemaExternal class (System.Xml.Schema)

ToString

### Description

An abstract class. Provides information about the included schema.

XmlSchemaExternal

*Example Syntax:*

ToString

[C#] protected XmlSchemaExternal();

[C++] protected: XmlSchemaExternal();

[VB] Protected Sub New()

[JScript] protected function XmlSchemaExternal();

Annotation

Id

LineNumber

LinePosition

Schema

ToString

### Description

Returns the **XmlSchema** for the referenced schema.

SchemaLocation

ToString

[C#] public string SchemaLocation {get; set;}

[C++] public: \_\_property String\* get\_SchemaLocation();public: \_\_property void  
set\_SchemaLocation(String\*);

[VB] Public Property SchemaLocation As String

[JScript] public function get SchemaLocation() : String;public function set  
SchemaLocation(String);

### *Description*

Provides the URI location for the schema, which tells the schema processor  
where the schema physically resides.

Imported schemas that do not have this attribute allow the imported  
schema's namespace to be determined by the XML document that is an instance of  
the containing schema or the application that is processing the schema.

SourceUri

UnhandledAttributes

XmlSchemaFacet class (System.Xml.Schema)

ToString

### *Description*

Abstract class for all facets that are used when simple types are extended by  
restriction.

A facet is defined as an element. Each **facet** element has a **fixed** attribute that is a Boolean value.

XmlSchemaFacet

*Example Syntax:*

ToString

[C#] protected XmlSchemaFacet();

[C++] protected: XmlSchemaFacet();

[VB] Protected Sub New()

[JScript] protected function XmlSchemaFacet();

Annotation

Id

IsFixed

ToString

### *Description*

Indicates that this facet is fixed.

If **True** , value is fixed. If **False** , value is not fixed.

LineNumber

LinePosition

SourceUri

UnhandledAttributes

Value

ToString

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25

*Description*

Provides the **value** attribute of the facet.  
 XmlSchemaForm enumeration (System.Xml.Schema)  
 ToString

*Description*

Indicates if **form** is qualified or unqualified.  
 ToString

[C#] public const XmlSchemaForm None;  
 [C++] public: const XmlSchemaForm None;  
 [VB] Public Const None As XmlSchemaForm  
 [JScript] public var None : XmlSchemaForm;

*Description*

Attribute can be qualified with the namespace prefix.  
 ToString

[C#] public const XmlSchemaForm Qualified;  
 [C++] public: const XmlSchemaForm Qualified;  
 [VB] Public Const Qualified As XmlSchemaForm  
 [JScript] public var Qualified : XmlSchemaForm;

*Description*

Attribute is not required to be qualified with the namespace prefix.

ToString

```
[C#] public const XmlSchemaForm Unqualified;
[C++] public: const XmlSchemaForm Unqualified;
[VB] Public Const Unqualified As XmlSchemaForm
[JScript] public var Unqualified : XmlSchemaForm;
```

*Description*

Attribute must be qualified with the namespace prefix.

XmlSchemaFractionDigitsFacet class (System.Xml.Schema)

ToString

*Description*

Class for defining **fractionDigits** facets.

XmlSchemaFractionDigitsFacet

*Example Syntax:*

ToString

```
[C#] public XmlSchemaFractionDigitsFacet();
[C++] public: XmlSchemaFractionDigitsFacet();
```



```

1  [VB] Public Sub New()
2  [JScript] public function XmlSchemaFractionDigitsFacet();
3      Annotation
4      Id
5      IsFixed
6      LineNumber
7      LinePosition
8      SourceUri
9      UnhandledAttributes
10     Value
11     XmlSchemaGroup class (System.Xml.Schema)
12     ToString

```

### *Description*

Class that defines groups at the **schema** level that are referenced from the complex types. Groups a set of element declarations so that they can be incorporated as a group into complex type definitions.

XmlSchemaGroup

*Example Syntax:*

ToString

```

23 [C#] public XmlSchemaGroup();
24 [C++] public: XmlSchemaGroup();

```

1 [VB] Public Sub New()

2 [JScript] public function XmlSchemaGroup();

3 Annotation

4 Id

5 LineNumber

6 LinePosition

7 Name

8 ToString

11 *Description*

12 Provides the name of the schema group.

13 The name must be an NCName as defined in the XML Namespaces  
14 specification.

15 Particle

16 ToString

18 [C#] public XmlSchemaGroupBase Particle {get; set;}

19 [C++] public: \_\_property XmlSchemaGroupBase\* get\_Particle();public:

20 \_\_property void set\_Particle(XmlSchemaGroupBase\*);

21 [VB] Public Property Particle As XmlSchemaGroupBase

22 [JScript] public function get Particle() : XmlSchemaGroupBase;public function set  
23 Particle(XmlSchemaGroupBase);

25 *Description*

One of **XmlSchemaChoice** , **XmlSchemaAll** , or **XmlSchemaSequence**  
classes.

SourceUri

UnhandledAttributes

XmlSchemaGroupBase class (System.Xml.Schema)

ToString

### *Description*

An abstract class for **XmlSchemaChoice** , **XmlSchemaAll** , or  
**XmlSchemaSequence** .

XmlSchemaGroupBase

*Example Syntax:*

ToString

[C#] protected XmlSchemaGroupBase();

[C++] protected: XmlSchemaGroupBase();

[VB] Protected Sub New()

[JScript] protected function XmlSchemaGroupBase();

Annotation

Id

Items

ToString

### Description

This collection is used to add new element types at the **schema** level.

LineNumber

LinePosition

MaxOccurs

MaxOccursString

MinOccurs

MinOccursString

SourceUri

UnhandledAttributes

XmlSchemaGroupRef class (System.Xml.Schema)

ToString

### Description

Class that defines groups at the **schema** level that are referenced from complex types. Groups a set of element declarations so that they can be incorporated as a group into complex type definitions.

XmlSchemaGroupRef

*Example Syntax:*

ToString

```
[C#] public XmlSchemaGroupRef();
```

```

1  [C++] public: XmlSchemaGroupRef();
2  [VB] Public Sub New()
3  [JScript] public function XmlSchemaGroupRef();
4      Annotation
5      Id
6      LineNumber
7      LinePosition
8      MaxOccurs
9      MaxOccursString
10     MinOccurs
11     MinOccursString
12     Particle
13     ToString
14
15
16     Description
17
18     RefName
19     ToString
20
21 [C#] public XmlQualifiedName RefName {get; set;}
22 [C++] public: __property XmlQualifiedName* get_RefName();public: __property
23 void set_RefName(XmlQualifiedName*);
24 [VB] Public Property RefName As XmlQualifiedName
25 [JScript] public function get RefName() : XmlQualifiedName;public function set

```

1 RefName(XmlQualifiedName);

2  
3 *Description*

4 Name of a group defined in this schema (or another schema indicated by  
5 the specified namespace).

6 The **RefName** value must be a QName. The **RefName** can include a  
7 namespace prefix. To include a group in a **complexType** element, use the  
8 **RefName** attribute to specify the group to include in this schema.

9 SourceUri

10 UnhandledAttributes

11 XmlSchemaIdentityConstraint class (System.Xml.Schema)

12 ToString

13  
14  
15 *Description*

16 Class for the identity constraints: **key** , **keyref** , **unique** elements.

17 An association between a name and one of several varieties of identity  
18 constraint related to uniqueness and reference.

19 XmlSchemaIdentityConstraint

20 *Example Syntax:*

21 ToString

22  
23 [C#] public XmlSchemaIdentityConstraint();

24 [C++] public: XmlSchemaIdentityConstraint();

[VB] Public Sub New()

[JScript] public function XmlSchemaIdentityConstraint();

Annotation

Fields

ToString

### *Description*

Collection of fields that apply as children for the XML Path Language (XPath) expression selector.

An identity constraint must contain one or more field elements. The collection specifies an **System.Xml.Schema.XmlSchemaXPath** expression that specifies the value (or one of the values) used to enforce an identity constraint ( **unique** , **key** , **keyref** ).

Id

LineNumber

LinePosition

Name

ToString

### *Description*

The name of the identity constraint.

Selector

ToString

```

1
2 [C#] public XmlSchemaXPath Selector {get; set;}
3 [C++] public: __property XmlSchemaXPath* get_Selector();public: __property
4 void set_Selector(XmlSchemaXPath*);
5 [VB] Public Property Selector As XmlSchemaXPath
6 [JScript] public function get Selector() : XmlSchemaXPath;public function set
7 Selector(XmlSchemaXPath);
8

```

### *Description*

The XML Path Language (XPath) expression **selector** element.

An XPath expression that is relative to the element being declared. This expression identifies the child elements that the identity constraint applies to.

SourceUri

UnhandledAttributes

XmlSchemaImport class (System.Xml.Schema)

ToString

### *Description*

The class to import schema components from any schema.

**XmlSchemaImport** allows references to schema components from other schemas with different target namespaces.

XmlSchemaImport

*Example Syntax:*

ToString



```

1
2 [C#] public XmlSchemaImport();
3 [C++] public: XmlSchemaImport();
4 [VB] Public Sub New()
5 [JScript] public function XmlSchemaImport();

```

6       Annotation

7       Id

8       LineNumber

9       LinePosition

10      Namespace

11      ToString

## 14 *Description*

15       Provides the target namespace for the imported schema as a URI reference.

16       The **namespace** attribute indicates that the containing schema document  
17 may contain qualified references to schema components in that namespace through  
18 one or more prefixes that are declared with **xmlns** attributes. If this attribute is  
19 absent, the containing schema can contain unqualified references to components in  
20 the imported namespace.

21       Schema

22       SchemaLocation

23       SourceUri

24       UnhandledAttributes

25       XmlSchemaInclude class (System.Xml.Schema)

ToString

*Description*

Class to include declarations and definitions from an external schema.

Allows them to be available for processing in the containing schema.

The included schema document must meet one of the following conditions:

The included schema must have the same target namespace as the containing schema document.

XmlSchemaInclude

*Example Syntax:*

ToString

[C#] public XmlSchemaInclude();

[C++] public: XmlSchemaInclude();

[VB] Public Sub New()

[JScript] public function XmlSchemaInclude();

Annotation

Id

LineNumber

LinePosition

Schema

SchemaLocation

SourceUri

UnhandledAttributes

XmlSchemaKey class (System.Xml.Schema)

ToString

### Description

Identifies a key constraint.

Specifies that an attribute or element value (or set of values) must be a key within the specified scope. A key must be unique, non-nillable/non-nullable, and always present.

XmlSchemaKey

*Example Syntax:*

ToString

[C#] public XmlSchemaKey();

[C++] public: XmlSchemaKey();

[VB] Public Sub New()

[JScript] public function XmlSchemaKey();

Annotation

Fields

Id

LineNumber

LinePosition

Name

Selector

SourceUri

UnhandledAttributes

XmlSchemaKeyref class (System.Xml.Schema)

ToString

### *Description*

Identifies a keyref constraint.

Specifies that an attribute or element value (or set of values) have a correspondence with those of the specified **key** or **unique** element. The name must be unique within a schema.

XmlSchemaKeyref

*Example Syntax:*

ToString

[C#] public XmlSchemaKeyref();

[C++] public: XmlSchemaKeyref();

[VB] Public Sub New()

[JScript] public function XmlSchemaKeyref();

Annotation

Fields

Id

LineNumber

LinePosition

Name

Refer

ToString

*Description*

The foreign key that this constraint refers to in another simple or complex type.

Name of a **key** or **unique** element defined in this schema (or another schema indicated by the specified namespace). The **refer** value must be a QName. The type can include a namespace prefix.

Selector

SourceUri

UnhandledAttributes

XmlSchemaLengthFacet class (System.Xml.Schema)

ToString

*Description*

Class for defining **length** facets.

XmlSchemaLengthFacet

*Example Syntax:*

ToString

[C#] public XmlSchemaLengthFacet();

[C++] public: XmlSchemaLengthFacet();

```

1 [VB] Public Sub New()
2 [JScript] public function XmlSchemaLengthFacet();
3     Annotation
4     Id
5     IsFixed
6     LineNumber
7     LinePosition
8     SourceUri
9     UnhandledAttributes
10    Value
11    XmlSchemaMaxExclusiveFacet class (System.Xml.Schema)
12    ToString

```

*Description*

Class for defining **maxExclusive** facets.

XmlSchemaMaxExclusiveFacet

*Example Syntax:*

ToString

```

21 [C#] public XmlSchemaMaxExclusiveFacet();
22 [C++] public: XmlSchemaMaxExclusiveFacet();
23 [VB] Public Sub New()
24 [JScript] public function XmlSchemaMaxExclusiveFacet();
25     Annotation

```

Id  
 IsFixed  
 LineNumber  
 LinePosition  
 SourceUri  
 UnhandledAttributes  
 Value  
 XmlSchemaMaxInclusiveFacet class (System.Xml.Schema)  
 ToString

### *Description*

Class for defining **maxInclusive** facets.

XmlSchemaMaxInclusiveFacet

*Example Syntax:*

ToString

[C#] public XmlSchemaMaxInclusiveFacet();

[C++] public: XmlSchemaMaxInclusiveFacet();

[VB] Public Sub New()

[JScript] public function XmlSchemaMaxInclusiveFacet();

Annotation

Id

IsFixed

LineNumber

LinePosition  
 SourceUri  
 UnhandledAttributes  
 Value  
 XmlSchemaMaxLengthFacet class (System.Xml.Schema)  
 ToString

*Description*

Class for defining **maxLength** facets.

XmlSchemaMaxLengthFacet

*Example Syntax:*

ToString

[C#] public XmlSchemaMaxLengthFacet();  
 [C++] public: XmlSchemaMaxLengthFacet();  
 [VB] Public Sub New()  
 [JScript] public function XmlSchemaMaxLengthFacet();

Annotation  
 Id  
 IsFixed  
 LineNumber  
 LinePosition  
 SourceUri  
 UnhandledAttributes



Value

XmlSchemaMinExclusiveFacet class (System.Xml.Schema)

ToString

### Description

Class for defining **minExclusive** facets.

XmlSchemaMinExclusiveFacet

*Example Syntax:*

ToString

[C#] public XmlSchemaMinExclusiveFacet();

[C++] public: XmlSchemaMinExclusiveFacet();

[VB] Public Sub New()

[JScript] public function XmlSchemaMinExclusiveFacet();

Annotation

Id

IsFixed

LineNumber

LinePosition

SourceUri

UnhandledAttributes

Value

XmlSchemaMinInclusiveFacet class (System.Xml.Schema)

ToString



Class for defining **minLength** facets.

XmlSchemaMinLengthFacet

*Example Syntax:*

ToString

[C#] public XmlSchemaMinLengthFacet();

[C++] public: XmlSchemaMinLengthFacet();

[VB] Public Sub New()

[JScript] public function XmlSchemaMinLengthFacet();

Annotation

Id

IsFixed

LineNumber

LinePosition

SourceUri

UnhandledAttributes

Value

XmlSchemaNotation class (System.Xml.Schema)

ToString

## Description

Class for the definition of a notation.

XmlSchemaNotation

*Example Syntax:*

```

1      ToString
2
3  [C#] public XmlSchemaNotation();
4  [C++] public: XmlSchemaNotation();
5  [VB] Public Sub New()
6  [JScript] public function XmlSchemaNotation();
7
8      Annotation
9
10     Id
11
12     LineNumber
13
14     LinePosition
15
16     Name
17
18     ToString
19
20
21
22
23
24
25

```

*Description*

The name of the notation.

Public

ToString

```

20 [C#] public string Public {get; set;}
21 [C++] public: __property String* get_Public();public: __property void
22 set_Public(String*);
23 [VB] Public Property Public As String
24 [JScript] public function get Public() : String;public function set Public(String);
25

```

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25

*Description*

The **public** attribute.  
  
SourceUri  
  
System  
  
ToString

*Description*

The **system** attribute.  
  
The default value is String.Empty.  
  
UnhandledAttributes  
  
XmlSchemaNumericFacet class (System.Xml.Schema)  
  
ToString

*Description*

Abstract class for defining **numeric** facets.  
  
XmlSchemaNumericFacet  
  
*Example Syntax:*  
  
ToString

[C#] protected XmlSchemaNumericFacet();  
[C++] protected: XmlSchemaNumericFacet();

1	[VB] Protected Sub New()
2	[JScript] protected function XmlSchemaNumericFacet();
3	Annotation
4	Id
5	IsFixed
6	LineNumber
7	LinePosition
8	SourceUri
9	UnhandledAttributes
10	Value
11	XmlSchemaObject class (System.Xml.Schema)
12	ToString
13	
14	
15	<i>Description</i>
16	Creates an empty schema.
17	XmlSchemaObject
18	<i>Example Syntax:</i>
19	ToString
20	
21	[C#] protected XmlSchemaObject();
22	[C++] protected: XmlSchemaObject();
23	[VB] Protected Sub New()
24	[JScript] protected function XmlSchemaObject();
25	LineNumber

```

1      ToString
2
3  [C#] public int LineNumber {get; set;}
4  [C++] public: __property int get_LineNumber();public: __property void
5  set_LineNumber(int);
6  [VB] Public Property LineNumber As Integer
7  [JScript] public function get LineNumber() : int;public function set
8  LineNumber(int);
9

```

#### Description

Line number in the file that the **schema** element refers to.

**LineNumber** is used to store the line number when a schema is read from file. This is reported through **System.Xml.Schema.XmlSchemaException** for error handling.

LinePosition

ToString

```

17
18 [C#] public int LinePosition {get; set;}
19 [C++] public: __property int get_LinePosition();public: __property void
20 set_LinePosition(int);
21 [VB] Public Property LinePosition As Integer
22 [JScript] public function get LinePosition() : int;public function set
23 LinePosition(int);
24

```

#### Description

Line position in the file that the **schema** element refers to.

This is used to store the line position when a schema is read from file. This is reported through **System.Xml.Schema.XmlSchemaException** for error handling.

SourceUri

ToString

[C#] public string SourceUri {get; set;}

[C++] public: \_\_property String\* get\_SourceUri();public: \_\_property void set\_SourceUri(String\*);

[VB] Public Property SourceUri As String

[JScript] public function get SourceUri() : String;public function set SourceUri(String);

### *Description*

Identifies the source location for the file that loaded the schema.

Provides information for exception handling.

XmlSchemaObjectCollection class (System.Xml.Schema)

ToString

### *Description*

An object collection class to handle **XmlSchemaObjects** when collections are returned from method calls.

XmlSchemaObjectCollection



*Example Syntax:*

ToString

[C#] public XmlSchemaObjectCollection();

[C++] public: XmlSchemaObjectCollection();

[VB] Public Sub New()

[JScript] public function XmlSchemaObjectCollection(); Initializes a new instance of the **XmlSchemaObjectCollection** class.

*Description*

Initializes a new instance of the **XmlSchemaObjectCollection** class.

XmlSchemaObjectCollection

*Example Syntax:*

ToString

[C#] public XmlSchemaObjectCollection(XmlSchemaObject parent);

[C++] public: XmlSchemaObjectCollection(XmlSchemaObject\* parent);

[VB] Public Sub New(ByVal parent As XmlSchemaObject)

[JScript] public function XmlSchemaObjectCollection(parent :  
XmlSchemaObject);

*Description*

Initializes a new instance of the **XmlSchemaObjectCollection** class that takes an **XmlSchemaObject** . XmlSchemaObject

Count

1 InnerList

2 Item

3 ToString

4  
5  
6 *Description*

7 Looks up the qualified name using the supplied **XmlQualifiedName** and  
8 returns the **XmlSchemaObject** -the XSD element for this qualified name. int

9 List

10 Add

11  
12 [C#] public int Add(XmlSchemaObject item);

13 [C++] public: int Add(XmlSchemaObject\* item);

14 [VB] Public Function Add(ByVal item As XmlSchemaObject) As Integer

15 [JScript] public function Add(item : XmlSchemaObject) : int;

16  
17 *Description*

18 Adds an **XmlSchemaObject** to the **XmlSchemaObjectCollection** .

19 XmlSchemaObject

20 Contains

21  
22 [C#] public bool Contains(XmlSchemaObject item);

23 [C++] public: bool Contains(XmlSchemaObject\* item);

24 [VB] Public Function Contains(ByVal item As XmlSchemaObject) As Boolean

25 [JScript] public function Contains(item : XmlSchemaObject) : Boolean;

1  
2 *Description*

3 Indicates if specified qualified name is located in the

4 **XmlSchemaObjectCollection** .

5 *Return Value:* Returns **true** if the specified qualified name is in the collection.

6 Otherwise returns **false** . If null is supplied then **false** is returned since there is no  
7 qualified name with a null name. XmlSchemaObject

8 **CopyTo**

9  
10 [C#] public void CopyTo(XmlSchemaObject[] array, int index);

11 [C++] public: void CopyTo(XmlSchemaObject\* array[], int index);

12 [VB] Public Sub CopyTo(ByVal array() As XmlSchemaObject, ByVal index As  
13 Integer)

14 [JScript] public function CopyTo(array : XmlSchemaObject[], index : int);

15

16 *Description*

17 Copies an **XmlSchemaObject** to the **XmlSchemaObjectCollection** .

18 **XmlSchemaObject[]** int

19 **GetEnumerator**

20

21 [C#] public new XmlSchemaObjectEnumerator GetEnumerator();

22 [C++] public: XmlSchemaObjectEnumerator\* GetEnumerator();

23 [VB] Shadows Public Function GetEnumerator() As

24 XmlSchemaObjectEnumerator

25 [JScript] public hide function GetEnumerator() : XmlSchemaObjectEnumerator;

## Description

Returns an enumerator for iterating through the **XmlSchemaObjects** contained in the **XmlSchemaObjectCollection** .

*Return Value:* The iterator returns **XmlSchemaObject** .

## IndexOf

[C#] public int IndexOf(XmlSchemaObject item);

[C++] public: int IndexOf(XmlSchemaObject\* item);

[VB] Public Function IndexOf(ByVal item As XmlSchemaObject) As Integer

[JScript] public function IndexOf(item : XmlSchemaObject) : int;

## Description

Gets the index of the specified **XmlSchemaObject** within the collection of the specified item. **XmlSchemaObject**

## Insert

[C#] public void Insert(int index, XmlSchemaObject item);

[C++] public: void Insert(int index, XmlSchemaObject\* item);

[VB] Public Sub Insert(ByVal index As Integer, ByVal item As XmlSchemaObject)

[JScript] public function Insert(index : int, item : XmlSchemaObject);

## Description

1 Inserts an **XmlSchemaObject** to the **XmlSchemaObjectCollection** .

2 **XmlSchemaObject**

3 **OnClear**

4  
5 [C#] protected override void OnClear();

6 [C++] protected: void OnClear();

7 [VB] Overrides Protected Sub OnClear()

8 [JScript] protected override function OnClear();

9  
10 *Description*

11 **OnClear** is invoked before the standard Clear behavior.

12 **OnInsert**

13  
14 [C#] protected override void OnInsert(int index, object item);

15 [C++] protected: void OnInsert(int index, Object\* item);

16 [VB] Overrides Protected Sub OnInsert(ByVal index As Integer, ByVal item As  
17 Object)

18 [JScript] protected override function OnInsert(index : int, item : Object);

19  
20 *Description*

21 **OnInsert** is invoked before the standard Insert behavior. int object

22 **OnRemove**

23  
24 [C#] protected override void OnRemove(int index, object item);

25 [C++] protected: void OnRemove(int index, Object\* item);

1 [VB] Overrides Protected Sub OnRemove(ByVal index As Integer, ByVal item  
2 As Object)

3 [JScript] protected override function OnRemove(index : int, item : Object);

4  
5 *Description*

6 **OnRemove** is invoked before the standard Remove behavior. int object

7 OnSet

8  
9 [C#] protected override void OnSet(int index, object oldValue, object newValue);

10 [C++] protected: void OnSet(int index, Object\* oldValue, Object\* newValue);

11 [VB] Overrides Protected Sub OnSet(ByVal index As Integer, ByVal oldValue As  
12 Object, ByVal newValue As Object)

13 [JScript] protected override function OnSet(index : int, oldValue : Object,  
14 newValue : Object);

15  
16 *Description*

17 **OnSet** is invoked before the standard Set behavior. int object object

18 Remove

19  
20 [C#] public void Remove(XmlSchemaObject item);

21 [C++] public: void Remove(XmlSchemaObject\* item);

22 [VB] Public Sub Remove(ByVal item As XmlSchemaObject)

23 [JScript] public function Remove(item : XmlSchemaObject);

24  
25 *Description*

Removes an **XmlSchemaObject** from the **XmlSchemaObjectCollection** .

**XmlSchemaObject**

**XmlSchemaObjectEnumerator** class (System.Xml.Schema)

**ToString**

*Description*

An enumerator class to walk the **XmlSchemaObjectCollection** collections.

**Current**

**ToString**

[C#] public XmlSchemaObject Current {get;}

[C++] public: \_\_property XmlSchemaObject\* get\_Current();

[VB] Public ReadOnly Property Current As XmlSchemaObject

[JScript] public function get Current() : XmlSchemaObject;

*Description*

Returns the **XmlSchemaObject** for the current enumerator's position.

**MoveNext**

[C#] public bool MoveNext();

[C++] public: bool MoveNext();

[VB] Public Function MoveNext() As Boolean

[JScript] public function MoveNext() : Boolean;

## Description

Moves to the next item in the collection.

*Return Value:* **False** at the end of the collection.

Reset

[C#] public void Reset();

[C++] public: void Reset();

[VB] Public Sub Reset()

[JScript] public function Reset();

## Description

Reset the enumerator to the start of the collection.

IEnumerator.MoveNext

[C#] bool IEnumerator.MoveNext();

[C++] bool IEnumerator::MoveNext();

[VB] Function MoveNext() As Boolean Implements IEnumerator.MoveNext

[JScript] function IEnumerator.MoveNext() : Boolean;

IEnumerator.Reset

[C#] void IEnumerator.Reset();

[C++] void IEnumerator::Reset();

[VB] Sub Reset() Implements IEnumerator.Reset

[JScript] function IEnumerator.Reset();



XmlSchemaObjectTable class (System.Xml.Schema)

ToString

### Description

A collection class that provides read-only helpers for **XmlSchemaObject** objects. This class is used to provide the collections for contained elements that are within the schema as collections that are accessed from the **XmlSchema** class (for example, **Attributes** , **AttributeGroups** , **Elements** , and so on).

Count

ToString

[C#] public int Count {get;}

[C++] public: \_\_property int get\_Count();

[VB] Public ReadOnly Property Count As Integer

[JScript] public function get Count() : int;

### Description

Count.

Item

ToString

[C#] public XmlSchemaObject this[XmlQualifiedName name] {get;}

[C++] public: \_\_property XmlSchemaObject\* get\_Item(XmlQualifiedName\* name);

```

1 [VB] Public Default ReadOnly Property Item(ByVal name As
2 XmlQualifiedName) As XmlSchemaObject
3 [JScript] returnValue = XmlSchemaObjectTableObject.Item(name);
4

```

#### *Description*

Looks up the qualified name by the supplied **XmlQualifiedName** and returns the **XmlSchemaObject** which is the XML Schema Definition language (XSD) element for this qualified name.

Returns null if there is qualified name associated with the given name.

Names

ToString

```

13 [C#] public ICollection Names {get;}

```

```

14 [C++] public: __property ICollection* get_Names();

```

```

15 [VB] Public ReadOnly Property Names As ICollection

```

```

16 [JScript] public function get Names() : ICollection;

```

#### *Description*

Returns a collection of all the named elements in the collection as **XmlSchemaObject**.

Values

ToString

```

24 [C#] public ICollection Values {get;}

```

```

25 [C++] public: __property ICollection* get_Values();

```

1 [VB] Public ReadOnly Property Values As ICollection

2 [JScript] public function get Values() : ICollection;

3  
4 *Description*

5 Returns a collection of the values for all the elements in the collection as

6 **XmlSchemaObject .**

7 Contains

8  
9 [C#] public bool Contains(XmlQualifiedName name);

10 [C++] public: bool Contains(XmlQualifiedName\* name);

11 [VB] Public Function Contains(ByVal name As XmlQualifiedName) As Boolean

12 [JScript] public function Contains(name : XmlQualifiedName) : Boolean;

13  
14 *Description*

15 Returns **true** if the specified qualified name is in the collection.

16 Returns **false** if a null value is supplied because there is not a qualified

17 name for a null name. For more information, see

18 **System.Xml.XmlQualifiedName .**

19 GetEnumerator

20  
21 [C#] public IDictionaryEnumerator GetEnumerator();

22 [C++] public: IDictionaryEnumerator\* GetEnumerator();

23 [VB] Public Function GetEnumerator() As IDictionaryEnumerator

24 [JScript] public function GetEnumerator() : IDictionaryEnumerator;

1  
2 *Description*

3 Returns an enumerator for the **XmlSchemaObject** contained in the  
4 collection.

5 *Return Value:* Returns the **XmlSchemaObject** objects.

6 XmlSchemaParticle class (System.Xml.Schema)

7 ToString

8  
9  
10 *Description*

11 Base class for all particles types.

12 Particle types are usually interchangeable. A local element declaration or  
13 reference to a global element declaration (element), a compositor ( **sequence** ,  
14 **choice** , or **all** ), a reference to a named content model group (group), or an  
15 element wildcard (any).

16 XmlSchemaParticle

17 *Example Syntax:*

18 ToString

19  
20 [C#] protected XmlSchemaParticle();

21 [C++] protected: XmlSchemaParticle();

22 [VB] Protected Sub New()

23 [JScript] protected function XmlSchemaParticle();

24 Annotation

25 Id

LineNumber

LinePosition

MaxOccurs

ToString

*Description*

Maximum number of times the particle can occur.

The value can be an integer greater than or equal to zero. The default value is 1 .

MaxOccursString

ToString

[C#] public string MaxOccursString {get; set;}

[C++] public: \_\_property String\* get\_MaxOccursString();public: \_\_property void set\_MaxOccursString(String\*);

[VB] Public Property MaxOccursString As String

[JScript] public function get MaxOccursString() : String;public function set MaxOccursString(String);

*Description*

Provides the number as a string value. Maximum number of times the particle can occur.

To set no limit on the maximum number, use the string "unbounded".

MinOccurs

## ToString

[C#] public decimal MinOccurs {get; set;}

[C++] public: \_\_property Decimal get\_MinOccurs();public: \_\_property void  
set\_MinOccurs(Decimal);

[VB] Public Property MinOccurs As Decimal

[JScript] public function get MinOccurs() : Decimal;public function set  
MinOccurs(Decimal);

### Description

Minimum number of times the particle can occur.

The value can be an integer greater than or equal to zero. Set this attribute  
to zero to indicate that the attribute is optional. The default value is **1** .

## MinOccursString

## ToString

[C#] public string MinOccursString {get; set;}

[C++] public: \_\_property String\* get\_MinOccursString();public: \_\_property void  
set\_MinOccursString(String\*);

[VB] Public Property MinOccursString As String

[JScript] public function get MinOccursString() : String;public function set  
MinOccursString(String);

### Description

Provides the number as a string value. Minimum number of times the particle can occur.

SourceUri

UnhandledAttributes

XmlSchemaPatternFacet class (System.Xml.Schema)

ToString

### *Description*

Class for defining **pattern** facets.

XmlSchemaPatternFacet

*Example Syntax:*

ToString

[C#] public XmlSchemaPatternFacet();

[C++] public: XmlSchemaPatternFacet();

[VB] Public Sub New()

[JScript] public function XmlSchemaPatternFacet();

Annotation

Id

IsFixed

LineNumber

LinePosition

SourceUri

UnhandledAttributes

Value

XmlSchemaRedefine class (System.Xml.Schema)

ToString

### *Description*

Allows simple and complex types, groups, and attribute groups from external schema files to be redefined in the current schema. This class provides versioning for the schema elements.

This class does the following: The **redefine** element requires that the external elements exist in the same target namespace as the redefining schema. Schemas without a namespace can also be redefined using the **redefine** element in which case, the redefined components become part of the schema's target namespace.

XmlSchemaRedefine

*Example Syntax:*

ToString

[C#] public XmlSchemaRedefine();

[C++] public: XmlSchemaRedefine();

[VB] Public Sub New()

[JScript] public function XmlSchemaRedefine();

Annotation

AttributeGroups

ToString



1  
2  
3 *Description*

4 The **XmlSchemaObjectTable** for all attributes in the schema.

5 Groups

6 ToString

7  
8 [C#] public XmlSchemaObjectTable Groups {get;}

9 [C++] public: \_\_property XmlSchemaObjectTable\* get\_Groups();

10 [VB] Public ReadOnly Property Groups As XmlSchemaObjectTable

11 [JScript] public function get Groups() : XmlSchemaObjectTable;

12  
13 *Description*

14 The **XmlSchemaObjectTable** for all groups in the schema.

15 Id

16 Items

17 ToString

18  
19  
20 *Description*

21 Collection of the following classes: **XmlSchemaAnnotation** ,

22 **XmlSchemaAttribute** , **XmlSchemaAttributeGroup** ,

23 **XmlSchemaComplexType** , **XmlSchemaSimpleType** , **XmlSchemaElement** ,

24 **XmlSchemaGroup** , and **XmlSchemaNotation** .

25 LineNumber

1       LinePosition  
2       Schema  
3       SchemaLocation  
4       SchemaTypes  
5       ToString

6  
7  
8       *Description*

9       The **XmlSchemaObjectTable** for all schema types in the schema.

10       SourceUri

11       UnhandledAttributes

12       XmlSchemaSequence class (System.Xml.Schema)

13       ToString

14  
15  
16       *Description*

17       Requires the elements in the group to appear in the specified sequence  
18       within the containing element. Represents the W3C **sequence** (compositor)  
19       element.

20       XmlSchemaSequence

21       *Example Syntax:*

22       ToString

23  
24       [C#] public XmlSchemaSequence();

25       [C++] public: XmlSchemaSequence();

1 [VB] Public Sub New()

2 [JScript] public function XmlSchemaSequence();

3 Annotation

4 Id

5 Items

6 ToString

7  
8  
9 *Description*

10 The elements contained within the compositor. Collection of  
11 **XmlSchemaElement** , **XmlSchemaGroupRef** , **XmlSchemaChoice** ,  
12 **XmlSchemaSequence** , or **XmlSchemaAny** .

13 LineNumber

14 LinePosition

15 MaxOccurs

16 MaxOccursString

17 MinOccurs

18 MinOccursString

19 SourceUri

20 UnhandledAttributes

21 XmlSchemaSimpleContent class (System.Xml.Schema)

22 ToString

23  
24  
25 *Description*

Class for simple types and complex types with a simple content model.

Contains extensions or restrictions on a complex type with character data or on a simple type as content and contains no elements.

XmlSchemaSimpleContent

*Example Syntax:*

ToString

[C#] public XmlSchemaSimpleContent();

[C++] public: XmlSchemaSimpleContent();

[VB] Public Sub New()

[JScript] public function XmlSchemaSimpleContent();

Annotation

Content

ToString

### *Description*

One of **XmlSchemaSimpleContentRestriction** or **XmlSchemaSimpleContentExtension** .

The **simpleContent** element enables you to specify an element as containing a simple type with no elements and restrict the value of the element's content or extend the content with attributes.

Id

LineNumber

LinePosition

SourceUri

UnhandledAttributes

XmlSchemaSimpleContentExtension class (System.Xml.Schema)

ToString

### Description

Class for simple types that are extended by extension. Extends the simple type content of the element by adding attributes.

XmlSchemaSimpleContentExtension

*Example Syntax:*

ToString

[C#] public XmlSchemaSimpleContentExtension();

[C++] public: XmlSchemaSimpleContentExtension();

[VB] Public Sub New()

[JScript] public function XmlSchemaSimpleContentExtension();

Annotation

AnyAttribute

ToString

### Description

Allows an **XmlSchemaAnyAttribute** to be used for the attribute value.

Attributes

ToString

[C#] public XmlSchemaObjectCollection Attributes {get;}

[C++] public: \_\_property XmlSchemaObjectCollection\* get\_Attributes();

[VB] Public ReadOnly Property Attributes As XmlSchemaObjectCollection

[JScript] public function get Attributes() : XmlSchemaObjectCollection;

### *Description*

Contains **XmlSchemaAttribute** and **XmlSchemaAttributeGroupRef**.

Collection of attributes for the **simpleType** element.

BaseTypeName

ToString

[C#] public XmlQualifiedName BaseTypeName {get; set;}

[C++] public: \_\_property XmlQualifiedName\* get\_BaseTypeName();public:

\_\_property void set\_BaseTypeName(XmlQualifiedName\*);

[VB] Public Property BaseTypeName As XmlQualifiedName

[JScript] public function get BaseTypeName() : XmlQualifiedName;public

function set BaseTypeName(XmlQualifiedName);

### *Description*

Name of a built-in data type, simple type, or complex type.

Id

LineNumber

LinePosition

SourceUri

UnhandledAttributes

XmlSchemaSimpleContentRestriction class (System.Xml.Schema)

ToString

### Description

Class for simple types that are extended by restriction. Restricts the range of values for the element to a subset of the inherited simple types.

XmlSchemaSimpleContentRestriction

*Example Syntax:*

ToString

[C#] public XmlSchemaSimpleContentRestriction();

[C++] public: XmlSchemaSimpleContentRestriction();

[VB] Public Sub New()

[JScript] public function XmlSchemaSimpleContentRestriction();

Annotation

AnyAttribute

ToString

### Description

Allows an **XmlSchemaAnyAttribute** to be used for the attribute value.

Attributes

```

1      ToString
2
3  [C#] public XmlSchemaObjectCollection Attributes {get;}
4  [C++] public: __property XmlSchemaObjectCollection* get_Attributes();
5  [VB] Public ReadOnly Property Attributes As XmlSchemaObjectCollection
6  [JScript] public function get Attributes() : XmlSchemaObjectCollection;
7
8  Description
9      Contains XmlSchemaAttribute and XmlSchemaAttributeGroupRef .
10     Collection of attributes for the simple type.
11
12     BaseType
13
14     ToString
15
16  [C#] public XmlSchemaSimpleType BaseType {get; set;}
17  [C++] public: __property XmlSchemaSimpleType* get_BaseType();public:
18  __property void set_BaseType(XmlSchemaSimpleType*);
19  [VB] Public Property BaseType As XmlSchemaSimpleType
20  [JScript] public function get BaseType() : XmlSchemaSimpleType;public function
21  set BaseType(XmlSchemaSimpleType);
22
23  Description
24      Derived from the type specified by the base value.
25      The base value must be a QName.
26
27      BaseTypeName
28
29      ToString

```



```

1
2 [C#] public XmlQualifiedName BaseTypeName {get; set;}
3 [C++] public: __property XmlQualifiedName* get_BaseTypeName();public:
4 __property void set_BaseTypeName(XmlQualifiedName*);
5 [VB] Public Property BaseTypeName As XmlQualifiedName
6 [JScript] public function get BaseTypeName() : XmlQualifiedName;public
7 function set BaseTypeName(XmlQualifiedName);

```

### *Description*

Name of the built-in data type, simple type, or complex type.

Facets

ToString

```

13
14 [C#] public XmlSchemaObjectCollection Facets {get;}
15 [C++] public: __property XmlSchemaObjectCollection* get_Facets();
16 [VB] Public ReadOnly Property Facets As XmlSchemaObjectCollection
17 [JScript] public function get Facets() : XmlSchemaObjectCollection;

```

### *Description*

One or more of the following classes: **XmlSchemaLengthFacet** ,  
**XmlSchemaMinLengthFacet** , **XmlSchemaMaxLengthFacet** ,  
**XmlSchemaPatternFacet** , **XmlSchemaEnumerationFacet** ,  
**XmlSchemaMaxInclusiveFacet** , **XmlSchemaMaxExclusiveFacet** ,  
**XmlSchemaMinInclusiveFacet** , **XmlSchemaMinExclusiveFacet** ,  
**XmlSchemaFractionDigitsFacet** , **XmlSchemaTotalDigitsFacet** .

1 Id  
 2 LineNumber  
 3 LinePosition  
 4 SourceUri  
 5 UnhandledAttributes  
 6 XmlSchemaSimpleType class (System.Xml.Schema)  
 7 ToString

10 *Description*

11 Class defines a simple type that determines the information and constraints  
 12 for the values of attributes or elements with text-only content.

13 Simple types are defined by deriving them from existing simple types  
 14 (built-in data types and derived simple types). A simple type cannot contain  
 15 elements and cannot have attributes.

16 XmlSchemaSimpleType

17 *Example Syntax:*

18 ToString

20 [C#] public XmlSchemaSimpleType();  
 21 [C++] public: XmlSchemaSimpleType();  
 22 [VB] Public Sub New()  
 23 [JScript] public function XmlSchemaSimpleType();

24 Annotation

25 BaseSchemaType

1	Content
2	ToString
3	
4	
5	<i>Description</i>
6	One of <b>XmlSchemaSimpleTypeUnion</b> , <b>XmlSchemaSimpleTypeList</b> , or
7	<b>XmlSchemaSimpleTypeRestriction</b> .
8	Simple types can be defined in one of the following ways.
9	Datatype
10	DerivedBy
11	Final
12	FinalResolved
13	Id
14	IsMixed
15	LineNumber
16	LinePosition
17	Name
18	QualifiedName
19	SourceUri
20	UnhandledAttributes
21	XmlSchemaSimpleTypeContent class (System.Xml.Schema)
22	ToString
23	
24	
25	<i>Description</i>

Abstract class for simple type content classes.

XmlSchemaSimpleTypeContent

*Example Syntax:*

ToString

[C#] protected XmlSchemaSimpleTypeContent();

[C++] protected: XmlSchemaSimpleTypeContent();

[VB] Protected Sub New()

[JScript] protected function XmlSchemaSimpleTypeContent();

Annotation

Id

LineNumber

LinePosition

SourceUri

UnhandledAttributes

XmlSchemaSimpleTypeList class (System.Xml.Schema)

ToString

### *Description*

Class for the list of **simpleType** elements. Defines a **simpleType** element as a list of values of a specified data type.

When a data type is derived from a list data type, the following constraining facets can be used: **length** , **maxLength** , and **minLength** enumerations.

XmlSchemaSimpleTypeList

*Example Syntax:*

ToString

[C#] public XmlSchemaSimpleTypeList();

[C++] public: XmlSchemaSimpleTypeList();

[VB] Public Sub New()

[JScript] public function XmlSchemaSimpleTypeList();

Annotation

Id

ItemType

ToString

#### *Description*

This **simpleType** element is derived from the type specified by the base value.

The base value must be a QName.

ItemTypeName

ToString

[C#] public XmlQualifiedName ItemTypeName {get; set;}

[C++] public: \_\_property XmlQualifiedName\* get\_ItemTypeName();public:

\_\_property void set\_ItemTypeName(XmlQualifiedName\*);

[VB] Public Property ItemTypeName As XmlQualifiedName

[JScript] public function get ItemTypeName() : XmlQualifiedName;public

1 function set ItemTypeName(XmlQualifiedName);

2  
3 *Description*

4 The type name of simple type list. Name of a built-in data type or  
5 **simpleType** element defined in this schema (or another schema indicated by the  
6 specified namespace).

7 The **simpleType** element containing the **list** element is derived from the  
8 simple type specified by the list value. The list value must be a QName.

9 LineNumber

10 LinePosition

11 SourceUri

12 UnhandledAttributes

13 XmlSchemaSimpleTypeRestriction class (System.Xml.Schema)

14 ToString

15  
16  
17 *Description*

18 Class for the restriction of **simpleType** elements.

19 XmlSchemaSimpleTypeRestriction

20 *Example Syntax:*

21 ToString

22  
23 [C#] public XmlSchemaSimpleTypeRestriction();

24 [C++] public: XmlSchemaSimpleTypeRestriction();

```

1  [VB] Public Sub New()
2  [JScript] public function XmlSchemaSimpleTypeRestriction();
3      Annotation
4      BaseType
5      ToString
6
7
8  Description
9      Provides the information on the base type.
10     BaseTypeName
11     ToString
12
13  [C#] public XmlQualifiedName BaseTypeName {get; set;}
14  [C++] public: __property XmlQualifiedName* get_BaseTypeName();public:
15  __property void set_BaseTypeName(XmlQualifiedName*);
16  [VB] Public Property BaseTypeName As XmlQualifiedName
17  [JScript] public function get BaseTypeName() : XmlQualifiedName;public
18  function set BaseTypeName(XmlQualifiedName);
19
20  Description
21      The name of the qualified base type.
22      Facets
23      ToString
24
25  [C#] public XmlSchemaObjectCollection Facets {get;}

```

```

1 [C++] public: __property XmlSchemaObjectCollection* get_Facets();
2 [VB] Public ReadOnly Property Facets As XmlSchemaObjectCollection
3 [JScript] public function get Facets() : XmlSchemaObjectCollection;

```

#### *Description*

One or more of the following classes: **XmlSchemaLengthFacet** ,  
**XmlSchemaMinLengthFacet** , **XmlSchemaMaxLengthFacet** ,  
**XmlSchemaPatternFacet** , **XmlSchemaEnumerationFacet** ,  
**XmlSchemaMaxInclusiveFacet** , **XmlSchemaMaxExclusiveFacet** ,  
**XmlSchemaMinInclusiveFacet** , **XmlSchemaMinExclusiveFacet** ,  
**XmlSchemaFractionDigitsFacet** , **XmlSchemaTotalDigitsFacet** .

Id

LineNumber

LinePosition

SourceUri

UnhandledAttributes

XmlSchemaSimpleTypeUnion class (System.Xml.Schema)

ToString

#### *Description*

Class for the union of **simpleType** elements. Defines a **simpleType** element as a list of values of a specified data type.

A **union** type enables an element or attribute value to be one or more instances of one type drawn from the union of multiple atomic and list types.



XmlSchemaSimpleTypeUnion

*Example Syntax:*

ToString

[C#] public XmlSchemaSimpleTypeUnion();

[C++] public: XmlSchemaSimpleTypeUnion();

[VB] Public Sub New()

[JScript] public function XmlSchemaSimpleTypeUnion();

Annotation

BaseTypes

ToString

*Description*

Collection of base types.

Id

LineNumber

LinePosition

MemberTypesSource

ToString

*Description*

List of members of built-in data types or **simpleType** elements defined in this schema (or another schema indicated by the specified namespace).

The **simpleType** element containing the **union** element is derived from the simple types specified by the union value. The values in **memberTypes** property must be QNames.

SourceUri

UnhandledAttributes

XmlSchemaTotalDigitsFacet class (System.Xml.Schema)

ToString

### *Description*

Class for defining **totalDigits** facets.

XmlSchemaTotalDigitsFacet

*Example Syntax:*

ToString

[C#] public XmlSchemaTotalDigitsFacet();

[C++] public: XmlSchemaTotalDigitsFacet();

[VB] Public Sub New()

[JScript] public function XmlSchemaTotalDigitsFacet();

Annotation

Id

IsFixed

LineNumber

LinePosition

SourceUri

1       UnhandledAttributes  
2       Value  
3       XmlSchemaType class (System.Xml.Schema)  
4       ToString

7       *Description*

8       The base class for all simple types and complex types.

9       XmlSchemaType

10      *Example Syntax:*

11      ToString

13      [C#] public XmlSchemaType();

14      [C++] public: XmlSchemaType();

15      [VB] Public Sub New()

16      [JScript] public function XmlSchemaType();

17      Annotation

18      BaseSchemaType

19      ToString

22      *Description*

23      Name of a built-in XSD data type, **simpleType** element, or **complexType**  
24      element.

This **complexType** element is derived from the type specified by the base value. The base value must be a QName.

Datatype

ToString

```
[C#] public XmlSchemaDatatype Datatype {get;}
```

```
[C++] public: __property XmlSchemaDatatype* get_Datatype();
```

```
[VB] Public ReadOnly Property Datatype As XmlSchemaDatatype
```

```
[JScript] public function get Datatype() : XmlSchemaDatatype;
```

#### *Description*

Indicates the data type.

DerivedBy

ToString

```
[C#] public XmlSchemaDerivationMethod DerivedBy {get;}
```

```
[C++] public: __property XmlSchemaDerivationMethod get_DerivedBy();
```

```
[VB] Public ReadOnly Property DerivedBy As XmlSchemaDerivationMethod
```

```
[JScript] public function get DerivedBy() : XmlSchemaDerivationMethod;
```

#### *Description*

Indicates the how elements can derive from this element.

The value can contain #all or a list that is a subset of extension/restriction/substitution.

Final

ToString

[C#] public XmlSchemaDerivationMethod Final {get; set;}

[C++] public: \_\_property XmlSchemaDerivationMethod get\_Final();public:

\_\_property void set\_Final(XmlSchemaDerivationMethod);

[VB] Public Property Final As XmlSchemaDerivationMethod

[JScript] public function get Final() : XmlSchemaDerivationMethod;public

function set Final(XmlSchemaDerivationMethod);

FinalResolved

ToString

[C#] public XmlSchemaDerivationMethod FinalResolved {get;}

[C++] public: \_\_property XmlSchemaDerivationMethod get\_FinalResolved();

[VB] Public ReadOnly Property FinalResolved As XmlSchemaDerivationMethod

[JScript] public function get FinalResolved() : XmlSchemaDerivationMethod;

### *Description*

The value after element has been compiled to post-schema info set. This value indicates how the type is restricted when the type is extended through restriction. If the type is marked as final then this type cannot be extended.

This value is either from the type itself or if not defined on the type taken from the **schema** element.

Id

IsMixed

ToString

*Description*

Indicates that this type has a mixed content model (character data is allowed to appear between the child elements of this **complexType** element).

Default is false. This virtual method is overridden in derived classes.

LineNumber

LinePosition

Name

ToString

*Description*

Name of the type.

The name must be an NCName as defined in the XML Namespaces specification. Optional. If specified, the name must be unique among all **simpleType** and **complexType** elements.

QualifiedName

ToString

```
[C#] public XmlQualifiedName QualifiedName {get;}
```

```
[C++] public: __property XmlQualifiedName* get_QualifiedName();
```

```
[VB] Public ReadOnly Property QualifiedName As XmlQualifiedName
```

```
[JScript] public function get QualifiedName() : XmlQualifiedName;
```

*Description*

Qualified name for the type built from the **Name** attribute of this type.

SourceUri

UnhandledAttributes

XmlSchemaUnique class (System.Xml.Schema)

ToString

*Description*

Identifies a unique constraint among a set of elements.

Specifies that an attribute or element value (or a combination of attribute or element values) must be unique within the specified scope. The name must be unique within a schema.

XmlSchemaUnique

*Example Syntax:*

ToString

[C#] public XmlSchemaUnique();

[C++] public: XmlSchemaUnique();

[VB] Public Sub New()

[JScript] public function XmlSchemaUnique();

Annotation

Fields

Id

1	LineNumber
2	LinePosition
3	Name
4	Selector
5	SourceUri
6	UnhandledAttributes
7	XmlSchemaUse enumeration (System.Xml.Schema)
8	ToString

*Description*

Indicator of how the attribute is used.

Optional.

ToString

[C#] public const XmlSchemaUse None;  
 [C++] public: const XmlSchemaUse None;  
 [VB] Public Const None As XmlSchemaUse  
 [JScript] public var None : XmlSchemaUse;

*Description*

Attribute has no value.

ToString

[C#] public const XmlSchemaUse Optional;



1 [C++] public: const XmlSchemaUse Optional;  
 2 [VB] Public Const Optional As XmlSchemaUse  
 3 [JScript] public var Optional : XmlSchemaUse;

4  
 5 *Description*

6 Attribute is optional and may have any value.

7 ToString

8  
 9 [C#] public const XmlSchemaUse Prohibited;  
 10 [C++] public: const XmlSchemaUse Prohibited;  
 11 [VB] Public Const Prohibited As XmlSchemaUse  
 12 [JScript] public var Prohibited : XmlSchemaUse;

13  
 14 *Description*

15 Attribute cannot be used.

16 ToString

17  
 18 [C#] public const XmlSchemaUse Required;  
 19 [C++] public: const XmlSchemaUse Required;  
 20 [VB] Public Const Required As XmlSchemaUse  
 21 [JScript] public var Required : XmlSchemaUse;

22  
 23 *Description*

24 Attribute must appear once.

25 XmlSchemaWhiteSpaceFacet class (System.Xml.Schema)

ToString

*Description*

Class for defining **whitespace** facets.

XmlSchemaWhiteSpaceFacet

*Example Syntax:*

ToString

[C#] public XmlSchemaWhiteSpaceFacet();

[C++] public: XmlSchemaWhiteSpaceFacet();

[VB] Public Sub New()

[JScript] public function XmlSchemaWhiteSpaceFacet();

Annotation

Id

IsFixed

LineNumber

LinePosition

SourceUri

UnhandledAttributes

Value

XmlSchemaXPath class (System.Xml.Schema)

ToString

*Description*

Class for XML Path Language (XPath) expressions.

XmlSchemaXPath

*Example Syntax:*

ToString

[C#] public XmlSchemaXPath();

[C++] public: XmlSchemaXPath();

[VB] Public Sub New()

[JScript] public function XmlSchemaXPath();

Annotation

Id

LineNumber

LinePosition

SourceUri

UnhandledAttributes

XPath

ToString

*Description*

Attribute for the XML Path Language (XPath) expression.

1 An XPath expression that is relative to each element that is selected by the  
2 selector of the identity constraint. This expression must identify a single element  
3 or attribute whose content or value is used for the constraint. If the expression  
4 identifies an element, that element must be a simple type.

5 XmlSeverityType enumeration (System.Xml.Schema)

6 ToString

7  
8  
9 *Description*

10 Represents the severity of the validation event.

11 ToString

12  
13 [C#] public const XmlSeverityType Error;

14 [C++] public: const XmlSeverityType Error;

15 [VB] Public Const Error As XmlSeverityType

16 [JScript] public var Error : XmlSeverityType; Errors that can be recovered from.

17  
18 *Description*

19 Indicates a validation error occurred when validating the instance  
20 document. This applies to DTDs and XML-Data Reduced Language (XDR) and  
21 XML Schema Definition language (XSD) schemas. The W3C validity constraints  
22 are considered fatal errors.

23 ToString

24  
25 [C#] public const XmlSeverityType Warning;

1 [C++] public: const XmlSeverityType Warning;

2 [VB] Public Const Warning As XmlSeverityTy

### 4 System.Xml.Serialization

5 The namespace contains classes that are used to serialize objects into XML  
6 format documents or streams.

#### 8 *Description*

9 The **System.Xml.Serialization** namespace contains classes that are used to  
10 serialize objects into XML format documents or streams.

11 CodeIdentifier class (System.Xml.Serialization)

12 Constructors:

13 CodeIdentifier

14 *Example Syntax:*

15 Methods:

16 MakeCamel

18 [C#] public static string MakeCamel(string identifier);

19 [C++] public: static String\* MakeCamel(String\* identifier);

20 [VB] Public Shared Function MakeCamel(ByVal identifier As String) As String

21 [JScript] public static function MakeCamel(identifier : String) : String;

#### 23 *Description*

25 MakePascal

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25

```
[C#] public static string MakePascal(string identifier);
[C++] public: static String* MakePascal(String* identifier);
[VB] Public Shared Function MakePascal(ByVal identifier As String) As String
[JScript] public static function MakePascal(identifier : String) : String;
```

*Description*

**MakeValid**

```
[C#] public static string MakeValid(string identifier);
[C++] public: static String* MakeValid(String* identifier);
[VB] Public Shared Function MakeValid(ByVal identifier As String) As String
[JScript] public static function MakeValid(identifier : String) : String;
```

*Description*

CodeIdentifiers class (System.Xml.Serialization)

ToString

CodeIdentifiers

*Example Syntax:*

ToString

Properties:

UseCamelCasing

ToString

1  
2  
3 *Description*

4  
5 Add

6  
7 [C#] public void Add(string identifier, object value);

8 [C++] public: void Add(String\* identifier, Object\* value);

9 [VB] Public Sub Add(ByVal identifier As String, ByVal value As Object)

10 [JScript] public function Add(identifier : String, value : Object);

11  
12 *Description*

13  
14 AddReserved

15  
16 [C#] public void AddReserved(string identifier);

17 [C++] public: void AddReserved(String\* identifier);

18 [VB] Public Sub AddReserved(ByVal identifier As String)

19 [JScript] public function AddReserved(identifier : String);

20  
21 *Description*

22  
23 AddUnique

24  
25 [C#] public string AddUnique(string identifier, object value);

1 [C++] public: String\* AddUnique(String\* identifier, Object\* value);  
 2 [VB] Public Function AddUnique(ByVal identifier As String, ByVal value As  
 3 Object) As String  
 4 [JScript] public function AddUnique(identifier : String, value : Object) : String;

5  
 6 *Description*

7  
 8 Clear

9  
 10 [C#] public void Clear();  
 11 [C++] public: void Clear();  
 12 [VB] Public Sub Clear()  
 13 [JScript] public function Clear();

14  
 15 *Description*

16  
 17 IsInUse

18  
 19 [C#] public bool IsInUse(string identifier);  
 20 [C++] public: bool IsInUse(String\* identifier);  
 21 [VB] Public Function IsInUse(ByVal identifier As String) As Boolean  
 22 [JScript] public function IsInUse(identifier : String) : Boolean;

23  
 24 *Description*

25



## MakeRightCase

```
[C#] public string MakeRightCase(string identifier);  
[C++] public: String* MakeRightCase(String* identifier);  
[VB] Public Function MakeRightCase(ByVal identifier As String) As String  
[JScript] public function MakeRightCase(identifier : String) : String;
```

### *Description*

## MakeUnique

```
[C#] public string MakeUnique(string identifier);  
[C++] public: String* MakeUnique(String* identifier);  
[VB] Public Function MakeUnique(ByVal identifier As String) As String  
[JScript] public function MakeUnique(identifier : String) : String;
```

### *Description*

## Remove

```
[C#] public void Remove(string identifier);  
[C++] public: void Remove(String* identifier);  
[VB] Public Sub Remove(ByVal identifier As String)  
[JScript] public function Remove(identifier : String);
```

*Description*

RemoveReserved

[C#] public void RemoveReserved(string identifier);  
[C++] public: void RemoveReserved(String\* identifier);  
[VB] Public Sub RemoveReserved(ByVal identifier As String)  
[JScript] public function RemoveReserved(identifier : String);

*Description*

ToArray

[C#] public object ToArray(Type type);  
[C++] public: Object\* ToArray(Type\* type);  
[VB] Public Function ToArray(ByVal type As Type) As Object  
[JScript] public function ToArray(type : Type) : Object;

*Description*

XmlSerializationReader.CollectionFixup class (System.Xml.Serialization)

ToString

1  
2  
3 *Description*

4  
5 XmlSerializationReader.CollectionFixup

6 *Example Syntax:*

7 ToString

8  
9 [C#] public XmlSerializationReader.CollectionFixup(object collection,  
10 XmlSerializationCollectionFixupCallback callback, object collectionItems);  
11 [C++] public: CollectionFixup(Object\* collection,  
12 XmlSerializationCollectionFixupCallback\* callback, Object\* collectionItems);  
13 [VB] Public Sub New(ByVal collection As Object, ByVal callback As  
14 XmlSerializationCollectionFixupCallback, ByVal collectionItems As Object)  
15 [JScript] public function XmlSerializationReader.CollectionFixup(collection :  
16 Object, callback : XmlSerializationCollectionFixupCallback, collectionItems :  
17 Object);  
18

19 *Description*

20  
21 Callback

22 ToString

23  
24 [C#] public XmlSerializationCollectionFixupCallback Callback {get;}

25 [C++] public: \_\_property XmlSerializationCollectionFixupCallback\*

```

1  get_Callback();
2  [VB] Public ReadOnly Property Callback As
3  XmlSerializationCollectionFixupCallback
4  [JScript] public function get Callback() :
5  XmlSerializationCollectionFixupCallback;

```

### *Description*

Collection

ToString

```

12 [C#] public object Collection {get;}
13 [C++] public: __property Object* get_Collection();
14 [VB] Public ReadOnly Property Collection As Object
15 [JScript] public function get Collection() : Object;

```

### *Description*

CollectionItems

ToString

```

22 [C#] public object CollectionItems {get;}
23 [C++] public: __property Object* get_CollectionItems();
24 [VB] Public ReadOnly Property CollectionItems As Object
25 [JScript] public function get CollectionItems() : Object;

```

1  
2 *Description*

3  
4 XmlSerializationReader.Fixup class (System.Xml.Serialization)

5 ToString  
6  
7

8 *Description*

9  
10 XmlSerializationReader.Fixup

11 *Example Syntax:*

12 ToString  
13

14 [C#] public XmlSerializationReader.Fixup(object o,

15 XmlSerializationFixupCallback callback, int count);

16 [C++] public: Fixup(Object\* o, XmlSerializationFixupCallback\* callback, int  
17 count);

18 [VB] Public Sub New(ByVal o As Object, ByVal callback As

19 XmlSerializationFixupCallback, ByVal count As Integer)

20 [JScript] public function XmlSerializationReader.Fixup(o : Object, callback :

21 XmlSerializationFixupCallback, count : int);  
22

23 *Description*  
24

25 XmlSerializationReader.Fixup

*Example Syntax:*

ToString

```
[C#] public XmlSerializationReader.Fixup(object o,
XmlSerializationFixupCallback callback, string[] ids);
[C++] public: Fixup(Object* o, XmlSerializationFixupCallback* callback, String*
ids __gc[]);
[VB] Public Sub New(ByVal o As Object, ByVal callback As
XmlSerializationFixupCallback, ByVal ids() As String)
[JScript] public function XmlSerializationReader.Fixup(o : Object, callback :
XmlSerializationFixupCallback, ids : String[]);
```

*Description*

Callback

ToString

```
[C#] public XmlSerializationFixupCallback Callback {get;}
[C++] public: __property XmlSerializationFixupCallback* get_Callback();
[VB] Public ReadOnly Property Callback As XmlSerializationFixupCallback
[JScript] public function get Callback() : XmlSerializationFixupCallback;
```

*Description*

Ids

ToString

[C#] public string[] Ids {get;}

[C++] public: \_\_property String\* get\_Ids();

[VB] Public ReadOnly Property Ids As String ()

[JScript] public function get Ids() : String[];

*Description*

Source

ToString

[C#] public object Source {get; set;}

[C++] public: \_\_property Object\* get\_Source();public: \_\_property void  
set\_Source(Object\*);

[VB] Public Property Source As Object

[JScript] public function get Source() : Object;public function set Source(Object);

*Description*

IXmlSerializable interface (System.Xml.Serialization)

ToString

*Description*

Provides a custom serialization format for a serializable object.

Examples of classes that implement this interface are

**System.Data.DataSet** and **System.Xml.XmlDocument** . This is different from the **System.Runtime.Serialization.ISerializable** interface in that it is totally XML-centric and stream based in the sense that large amounts of XML can be incrementally streamed through the **System.Data.DataSet.ReadXml** and **System.Data.DataSet.WriteXml** methods.

GetSchema

[C#] XmlSchema GetSchema();

[C++] XmlSchema\* GetSchema();

[VB] Function GetSchema() As XmlSchema

[JScript] function GetSchema() : XmlSchema;

*Return Value:* An **System.Xml.Schema.XmlSchema** object that represents the XML schema.

This optional method is used by webserviceutil to create a callable web service client class from the returned **System.Xml.Schema.XmlSchema** object.

This method is not needed to do simple runtime XML serialization.

ReadXml

[C#] void ReadXml(XmlReader reader);

[C++] void ReadXml(XmlReader\* reader);

[VB] Sub ReadXml(ByVal reader As XmlReader)

[JScript] function ReadXml(reader : XmlReader);



## Description

Converts an XML document into an object using the specified reader.

The reader is positioned within the stream at the point that you began writing (in

**System.Xml.Serialization.IXmlSerializable.WriteXml(System.Xml.XmlWriter)** ). It is your responsibility to read everything you wrote, no less and no more.

The **System.Xml.XmlReader** used to read the XML document.

### WriteXml

[C#] void WriteXml(XmlWriter writer);

[C++] void WriteXml(XmlWriter\* writer);

[VB] Sub WriteXml(ByVal writer As XmlWriter)

[JScript] function WriteXml(writer : XmlWriter);

## Description

Converts the serializable members of an object into an XML document.

You must write enough information to be able to implement the

**System.Xml.Serialization.IXmlSerializable.ReadXml(System.Xml.XmlReader)** method correctly. For example, if you are writing a list of elements, you provide a way to know how many you wrote so that

**System.Xml.Serialization.IXmlSerializable.ReadXml(System.Xml.XmlReader)** knows how when to stop reading. There are three ways to do this: Method

Example Write an initial count in the XML-instance document. The **XmlWriter** used to write the XML-document instance.

SoapAttributeAttribute class (System.Xml.Serialization)

WriteXml

### *Description*

Specifies that the **System.Xml.Serialization.XmlSerializer** should serialize the class member as a SOAP attribute.

Apply the **System.Xml.Serialization.SoapAttributeAttribute** to a public field to specify that the **System.Xml.Serialization.XmlSerializer** serializes the field as a SOAP attribute. You can specify an alternative name of the attribute by setting the

**System.Xml.Serialization.SoapAttributeAttribute.AttributeName** property.

Set the **System.Xml.Serialization.SoapAttributeAttribute.DataType** if the attribute must be given a specific type different from the field's type. If the attribute belongs to a specific XML namespace, set the

**System.Xml.Serialization.SoapAttributeAttribute.Namespace** property.

SoapAttributeAttribute

### *Example Syntax:*

WriteXml

[C#] public SoapAttributeAttribute();

[C++] public: SoapAttributeAttribute();

[VB] Public Sub New()

[JScript] public function SoapAttributeAttribute(); Initializes a new instance of the

**System.Xml.Serialization.SoapAttributeAttribute** class.

## Description

Initializes a new instance of the **System.Xml.Serialization.SoapAttributeAttribute** class.

SoapAttributeAttribute

*Example Syntax:*

WriteXml

[C#] public SoapAttributeAttribute(string attrName);

[C++] public: SoapAttributeAttribute(String\* attrName);

[VB] Public Sub New(ByVal attrName As String)

[JScript] public function SoapAttributeAttribute(attrName : String);

## Description

Initializes a new instance of the **System.Xml.Serialization.SoapAttributeAttribute** class using the specified value as the name of the XML attribute. The name of the XML attribute.

AttributeName

WriteXml

[C#] public string AttributeName {get; set;}

[C++] public: \_\_property String\* get\_AttributeName();public: \_\_property void set\_AttributeName(String\*);

[VB] Public Property AttributeName As String

[JScript] public function get AttributeName() : String;public function set

1 AttributeName(String);

3 *Description*

4 Gets or sets the name of the SOAP attribute generated by the

5 **System.Xml.Serialization.XmlSerializer** .

6     DataType

7     WriteXml

9 [C#] public string DataType {get; set;}

10 [C++] public: \_\_property String\* get\_DataType();public: \_\_property void

11 set\_DataType(String\*);

12 [VB] Public Property DataType As String

13 [JScript] public function get DataType() : String;public function set

14 DataType(String);

16 *Description*

17 Gets or sets the XML data type of the SOAP attribute generated by the

18 **System.Xml.Serialization.XmlSerializer** .

19     Namespace

20     WriteXml

22 [C#] public string Namespace {get; set;}

23 [C++] public: \_\_property String\* get\_Namespace();public: \_\_property void

24 set\_Namespace(String\*);

25 [VB] Public Property Namespace As String

[JScript] public function get Namespace() : String;public function set  
Namespace(String);

#### *Description*

The XML namespace of the SOAP attribute generated by the  
**System.Xml.Serialization.XmlSerializer** .

TypeId

SoapAttributeOverrides class (System.Xml.Serialization)

ToString

#### *Description*

Allows you to override property, field, and class attributes when you use  
the **System.Xml.Serialization.XmlSerializer** to serialize or deserialize an object  
using the SOAP protocol.

SoapAttributeOverrides

*Example Syntax:*

ToString

[C#] public SoapAttributeOverrides();

[C++] public: SoapAttributeOverrides();

[VB] Public Sub New()

[JScript] public function SoapAttributeOverrides();

Item

ToString

1 [C#] public SoapAttributes this[Type type] {get;}

2 [C++] public: \_\_property SoapAttributes\* get\_Item(Type\* type);

3 [VB] Public Default ReadOnly Property Item(ByVal type As Type) As

4 SoapAttributes

5 [JScript] returnValue = SoapAttributeOverridesObject.Item(type); Gets an object  
6 that represents the collection of overriding SOAP attributes.  
7

### 8 *Description*

9 Gets the object associated with the specified (base-class) type. The base-  
10 class **System.Type** that is associated with the collection of attributes you want to  
11 override.  
12

13 Item

14 ToString

15  
16 [C#] public SoapAttributes this[Type type, string member] {get;}

17 [C++] public: \_\_property SoapAttributes\* get\_Item(Type\* type, String\* member);

18 [VB] Public Default ReadOnly Property Item(ByVal type As Type, ByVal

19 member As String) As SoapAttributes

20 [JScript] returnValue = SoapAttributeOverridesObject.Item(type, member);  
21

### 22 *Description*

23 Gets the object associated with the specified (base-class) type. The member  
24 parameter specifies the base-class member that is overridden. The base-class class  
25 **System.Type** that is associated with the collection of attributes you want to

1 override. The name of the overridden member that specifies the

2 **System.Xml.Serialization.SoapAttributes** to return.

3 Add

4  
5 [C#] public void Add(Type type, SoapAttributes attributes);

6 [C++] public: void Add(Type\* type, SoapAttributes\* attributes);

7 [VB] Public Sub Add(ByVal type As Type, ByVal attributes As SoapAttributes)

8 [JScript] public function Add(type : Type, attributes : SoapAttributes); Adds a

9 **System.Xml.Serialization.SoapAttributes** object to the collection of collection

10 of **System.Xml.Serialization.SoapAttributes** objects.

11  
12 *Description*

13 Adds a **System.Xml.Serialization.SoapAttributes** object to the collection  
14 of collection of **System.Xml.Serialization.SoapAttributes** objects. The *type*  
15 parameter specifies an object to be overridden by the

16 **System.Xml.Serialization.SoapAttributes** object. The **System.Type** of the  
17 object that will be overridden. A **System.Xml.Serialization.SoapAttributes**  
18 object that represents the overriding attributes.

19 Add

20  
21 [C#] public void Add(Type type, string member, SoapAttributes attributes);

22 [C++] public: void Add(Type\* type, String\* member, SoapAttributes\* attributes);

23 [VB] Public Sub Add(ByVal type As Type, ByVal member As String, ByVal  
24 attributes As SoapAttributes)

25 [JScript] public function Add(type : Type, member : String, attributes :

SoapAttributes);

### Description

Adds a **System.Xml.Serialization.SoapAttributes** object to the collection of collection of **System.Xml.Serialization.SoapAttributes** objects. The *type* parameter specifies an object to be overridden by the **System.Xml.Serialization.SoapAttributes** object. The *member* parameter specifies the name of a member that will be overridden.

The **System.Xml.Serialization.SoapAttributes** object contains a union of attribute objects that cause the **System.Xml.Serialization.XmlSerializer** to override its default serialization behavior for a set of objects. You choose the attribute objects to place in the **System.Xml.Serialization.SoapAttributes** object, depending on the particular behaviors you want to override. For example, the **System.Xml.Serialization.XmlSerializer** serializes a class member as an XML element by default. If you want the member to be serialized as an **System.Xml.Serialization.SoapAttribute** instead, you would create an **System.Xml.Serialization.SoapAttributeAttribute** , assign it to the **System.Xml.Serialization.SoapAttributes.SoapAttribute** property of an **System.Xml.Serialization.SoapAttributes** , and add the **System.Xml.Serialization.SoapAttributes** object to the **System.Xml.Serialization.XmlAttributeOverrides** object. The **System.Type** of the object to override. The name of the member to override. An **System.Xml.Serialization.SoapAttributes** object that represents the overriding attributes.

SoapAttributes class (System.Xml.Serialization)



ToString

*Description*

Represents a collection of attribute objects that control how the **System.Xml.Serialization.XmlSerializer** serializes and deserializes SOAP methods.

SoapAttributes

*Example Syntax:*

ToString

[C#] public SoapAttributes();

[C++] public: SoapAttributes();

[VB] Public Sub New()

[JScript] public function SoapAttributes(); Initializes a new instance of the **System.Xml.Serialization.SoapAttributes** class.

*Description*

Initializes a new instance of the **System.Xml.Serialization.SoapAttributes** class.

SoapAttributes

*Example Syntax:*

ToString

[C#] public SoapAttributes(ICustomAttributeProvider provider);

```

1 [C++] public: SoapAttributes(ICustomAttributeProvider* provider);
2 [VB] Public Sub New(ByVal provider As ICustomAttributeProvider)
3 [JScript] public function SoapAttributes(provider : ICustomAttributeProvider);
4

```

#### *Description*

Initializes a new instance of the **System.Xml.Serialization.SoapAttributes** class using the specified custom type. Any object that implements the **System.Reflection.ICustomAttributeProvider** interface, such as the **System.Type** class.

SoapAttribute

ToString

```

13 [C#] public SoapAttributeAttribute SoapAttribute {get; set;}
14 [C++] public: __property SoapAttributeAttribute* get_SoapAttribute();public:
15 __property void set_SoapAttribute(SoapAttributeAttribute*);
16 [VB] Public Property SoapAttribute As SoapAttributeAttribute
17 [JScript] public function get SoapAttribute() : SoapAttributeAttribute;public
18 function set SoapAttribute(SoapAttributeAttribute);
19

```

#### *Description*

Gets or set sthe **System.Xml.Serialization.SoapAttributeAttribute** to override.

SoapDefaultValue

ToString

```

1
2 [C#] public object SoapDefaultValue {get; set;}
3 [C++] public: __property Object* get_SoapDefaultValue();public: __property
4 void set_SoapDefaultValue(Object*);
5 [VB] Public Property SoapDefaultValue As Object
6 [JScript] public function get SoapDefaultValue() : Object;public function set
7 SoapDefaultValue(Object);
8

```

### *Description*

Gets or sets the default value of an XML element or attribute.

SoapElement

ToString

```

13
14 [C#] public SoapElementAttribute SoapElement {get; set;}
15 [C++] public: __property SoapElementAttribute* get_SoapElement();public:
16 __property void set_SoapElement(SoapElementAttribute*);
17 [VB] Public Property SoapElement As SoapElementAttribute
18 [JScript] public function get SoapElement() : SoapElementAttribute;public
19 function set SoapElement(SoapElementAttribute);
20

```

### *Description*

Gets or sets a **System.Xml.Serialization.SoapElementAttribute** to override.

SoapEnum

ToString

```

1
2 [C#] public SoapEnumAttribute SoapEnum {get; set;}
3 [C++] public: __property SoapEnumAttribute* get_SoapEnum();public:
4 __property void set_SoapEnum(SoapEnumAttribute*);
5 [VB] Public Property SoapEnum As SoapEnumAttribute
6 [JScript] public function get SoapEnum() : SoapEnumAttribute;public function set
7 SoapEnum(SoapEnumAttribute);

```

#### 9 *Description*

10 Gets or sets an object that specifies how the  
11 **System.Xml.Serialization.XmlSerializer** serializes a SOAP enumeration.

12 SoapIgnore

13 ToString

```

14
15 [C#] public bool SoapIgnore {get; set;}
16 [C++] public: __property bool get_SoapIgnore();public: __property void
17 set_SoapIgnore(bool);
18 [VB] Public Property SoapIgnore As Boolean
19 [JScript] public function get SoapIgnore() : Boolean;public function set
20 SoapIgnore(Boolean);

```

#### 22 *Description*

23 Gets or sets a value that specifies whether the XmlSerializer serializes a  
24 public field or property with SOAP.

25 SoapType

```

1      ToString
2
3  [C#] public SoapTypeAttribute SoapType {get; set;}
4  [C++] public: __property SoapTypeAttribute* get_SoapType();public: __property
5  void set_SoapType(SoapTypeAttribute*);
6  [VB] Public Property SoapType As SoapTypeAttribute
7  [JScript] public function get SoapType() : SoapTypeAttribute;public function set
8  SoapType(SoapTypeAttribute);
9

```

#### *Description*

Gets or sets the SOAP attribute type.

SoapCodeExporter class (System.Xml.Serialization)

ToString

#### *Description*

SoapCodeExporter

*Example Syntax:*

ToString

```

22 [C#] public SoapCodeExporter(CodeNamespace codeNamespace);
23 [C++] public: SoapCodeExporter(CodeNamespace* codeNamespace);
24 [VB] Public Sub New(ByVal codeNamespace As CodeNamespace)
25 [JScript] public function SoapCodeExporter(codeNamespace : CodeNamespace);

```

*Description*

SoapCodeExporter

*Example Syntax:*

ToString

```
[C#] public SoapCodeExporter(CodeNamespace codeNamespace,
CodeCompileUnit codeCompileUnit);
[C++] public: SoapCodeExporter(CodeNamespace* codeNamespace,
CodeCompileUnit* codeCompileUnit);
[VB] Public Sub New(ByVal codeNamespace As CodeNamespace, ByVal
codeCompileUnit As CodeCompileUnit)
[JScript] public function SoapCodeExporter(codeNamespace : CodeNamespace,
codeCompileUnit : CodeCompileUnit);
```

IncludeMetadata

ToString

```
[C#] public CodeAttributeDeclarationCollection IncludeMetadata {get;}
[C++] public: __property CodeAttributeDeclarationCollection*
get_IncludeMetadata();
[VB] Public ReadOnly Property IncludeMetadata As
CodeAttributeDeclarationCollection
[JScript] public function get IncludeMetadata() :
CodeAttributeDeclarationCollection;
```

*Description*

**AddMappingMetadata**

[C#] public void AddMappingMetadata(CodeAttributeDeclarationCollection metadata, XmlMemberMapping member);

[C++] public: void AddMappingMetadata(CodeAttributeDeclarationCollection\* metadata, XmlMemberMapping\* member);

[VB] Public Sub AddMappingMetadata(ByVal metadata As CodeAttributeDeclarationCollection, ByVal member As XmlMemberMapping)

[JScript] public function AddMappingMetadata(metadata : CodeAttributeDeclarationCollection, member : XmlMemberMapping);

*Description*

**AddMappingMetadata**

[C#] public void AddMappingMetadata(CodeAttributeDeclarationCollection metadata, XmlMemberMapping member, bool forceUseMemberName);

[C++] public: void AddMappingMetadata(CodeAttributeDeclarationCollection\* metadata, XmlMemberMapping\* member, bool forceUseMemberName);

[VB] Public Sub AddMappingMetadata(ByVal metadata As CodeAttributeDeclarationCollection, ByVal member As XmlMemberMapping, ByVal forceUseMemberName As Boolean)

1 [JScript] public function AddMappingMetadata(metadata :  
2 CodeAttributeDeclarationCollection, member : XmlMemberMapping,  
3 forceUseMemberName : Boolean);

4  
5 *Description*

6  
7 ExportMembersMapping

8  
9 [C#] public void ExportMembersMapping(XmlMembersMapping  
10 xmlMembersMapping);

11 [C++] public: void ExportMembersMapping(XmlMembersMapping\*  
12 xmlMembersMapping);

13 [VB] Public Sub ExportMembersMapping(ByVal xmlMembersMapping As  
14 XmlMembersMapping)

15 [JScript] public function ExportMembersMapping(xmlMembersMapping :  
16 XmlMembersMapping);

17  
18 *Description*

19  
20 ExportTypeMapping

21  
22 [C#] public void ExportTypeMapping(XmlTypeMapping xmlTypeMapping);

23 [C++] public: void ExportTypeMapping(XmlTypeMapping\* xmlTypeMapping);

24 [VB] Public Sub ExportTypeMapping(ByVal xmlTypeMapping As  
25 XmlTypeMapping)



1 [JScript] public function ExportTypeMapping(xmlTypeMapping :  
2 XmlTypeMapping);

3  
4 *Description*

5  
6 SoapElementAttribute class (System.Xml.Serialization)

7 ToString

8  
9  
10 *Description*

11 Specifies that the public member value be serialized by the  
12 **System.Xml.Serialization.XmlSerializer** using the Simple Access Object  
13 Protocol (SOAP) as a SOAP element.

14 The SoapElementAttribute is used in two cases. First, it can be applied to  
15 public member of a class when creating an .asmx file. Second, you can apply the  
16 attribute to members in a class that is serialized by an XmlSerializer object.

17 SoapElementAttribute

18 *Example Syntax:*

19 ToString

20  
21 [C#] public SoapElementAttribute();

22 [C++] public: SoapElementAttribute();

23 [VB] Public Sub New()

24 [JScript] public function SoapElementAttribute(); Initializes a new instance of the

25 **System.Xml.Serialization.SoapElementAttribute** class.

1  
2 *Description*

3       Initializes a new instance of the  
4 **System.Xml.Serialization.SoapElementAttribute** class.

5       SoapElementAttribute

6       *Example Syntax:*

7       ToString

8  
9 [C#] public SoapElementAttribute(string elementName);  
10 [C++] public: SoapElementAttribute(String\* elementName);  
11 [VB] Public Sub New(ByVal elementName As String)  
12 [JScript] public function SoapElementAttribute(elementName : String);

13  
14 *Description*

15       Initializes a new instance of the  
16 **System.Xml.Serialization.SoapElementAttribute** class and species the name of  
17 the XML element. The XML-element name of the serialized member.

18       DataType

19       ToString

20  
21 [C#] public string DataType {get; set;}  
22 [C++] public: \_\_property String\* get\_DataType();public: \_\_property void  
23 set\_DataType(String\*);  
24 [VB] Public Property DataType As String  
25 [JScript] public function get DataType() : String;public function set

1   DataType(String);

3   *Description*

4       Gets or sets the XML data type of the generated XML element.

5       The following table lists the XML data types with their .NET equivalents.

6   For the XML **date** and **time** data types, the .NET data type consists of a

7   **System.DateTime** to which the

8   **System.Xml.Serialization.SoapElementAttribute** has been applied with the

9   **System.Xml.Serialization.SoapElementAttribute.DataType** set to "date" and

10   "time", respectively.

11       ElementName

12       ToString

14   [C#] public string ElementName {get; set;}

15   [C++] public: \_\_property String\* get\_ElementName();public: \_\_property void

16   set\_ElementName(String\*);

17   [VB] Public Property ElementName As String

18   [JScript] public function get ElementName() : String;public function set

19   ElementName(String);

21   *Description*

22       Gets or sets the name of the generated XML element.

23       Nullable

24       ToString

```

1
2 [C#] public bool IsNullable {get; set;}
3 [C++] public: __property bool get_IsNullable();public: __property void
4 set_IsNullable(bool);
5 [VB] Public Property IsNullable As Boolean
6 [JScript] public function get IsNullable() : Boolean;public function set
7 IsNullable(Boolean);
8     TypeId
9     SoapEnumAttribute class (System.Xml.Serialization)
10    ToString
11
12
13 Description
14     Controls how the System.Xml.Serialization.XmlSerializer serializes an
15 enumeration member.
16     SoapEnumAttribute
17     Example Syntax:
18     ToString
19
20 [C#] public SoapEnumAttribute();
21 [C++] public: SoapEnumAttribute();
22 [VB] Public Sub New()
23 [JScript] public function SoapEnumAttribute(); Initializes a new instance of the
24 System.Xml.Serialization.SoapEnumAttribute class.
25

```

1  
2 *Description*

3       Initializes a new instance of the  
4 **System.Xml.Serialization.SoapEnumAttribute** class.

5       SoapEnumAttribute

6       *Example Syntax:*

7       ToString

8  
9 [C#] public SoapEnumAttribute(string name);  
10 [C++] public: SoapEnumAttribute(String\* name);  
11 [VB] Public Sub New(ByVal name As String)  
12 [JScript] public function SoapEnumAttribute(name : String);  
13

14 *Description*

15       Initializes a new instance of the  
16 **System.Xml.Serialization.SoapEnumAttribute** class, using the specified  
17 element name. The XML element name generated by the  
18 **System.Xml.Serialization.XmlSerializer**.

19       Name

20       ToString

21  
22 [C#] public string Name {get; set;}  
23 [C++] public: \_\_property String\* get\_Name();public: \_\_property void  
24 set\_Name(String\*);  
25 [VB] Public Property Name As String

1 [JScript] public function get Name() : String;public function set Name(String);

3 *Description*

4 Gets or sets the value generated in an XML-document instance when the  
5 **System.Xml.Serialization.XmlSerializer** serializes an enumeration, or the value  
6 recognized when it deserializes the enumeration member.

7 TypeId

8 SoapIgnoreAttribute class (System.Xml.Serialization)

9 ToString

12 *Description*

13 Instructs the **System.Xml.Serialization.XmlSerializer** not to serialize the  
14 public field or public read/write property value.

15 The **System.Xml.Serialization.SoapIgnoreAttribute** belongs to a family  
16 of attributes that controls how the **System.Xml.Serialization.XmlSerializer** .  
17 serializes, or deserializes, an object. If you apply the  
18 **System.Xml.Serialization.SoapIgnoreAttribute** to any member of a class, the  
19 **System.Xml.Serialization.XmlSerializer** ignores the member when serializing or  
20 deserializing an instance of the class. For a complete list of similar attributes, see  
21 the **System.Xml.Serialization.XmlSerializer** class.

22 SoapIgnoreAttribute

23 *Example Syntax:*

24 ToString

```

1  [C#] public SoapIgnoreAttribute();
2
3  [C++] public: SoapIgnoreAttribute();
4
5  [VB] Public Sub New()
6
7  [JScript] public function SoapIgnoreAttribute();

```

### *Description*

Initializes a new instance of the **System.Xml.Serialization.SoapIgnoreAttribute** class.

TypeId

SoapIncludeAttribute class (System.Xml.Serialization)

ToString

### *Description*

Allows the **System.Xml.Serialization.XmlSerializer** to recognize a derived class when it serializes or deserializes an object.

SoapIncludeAttribute

*Example Syntax:*

ToString

```

22 [C#] public SoapIncludeAttribute(Type type);
23
24 [C++] public: SoapIncludeAttribute(Type* type);
25
26 [VB] Public Sub New(ByVal type As Type)
27
28 [JScript] public function SoapIncludeAttribute(type : Type);

```

1  
2 *Description*

3       Initializes a new instance of the  
4 **System.Xml.Serialization.SoapIncludeAttribute** class using the specified type.  
5 The **System.Type** of the derived class to include.

6       Type

7       ToString

8  
9 [C#] public Type Type {get; set;}

10 [C++] public: \_\_property Type\* get\_Type();public: \_\_property void  
11 set\_Type(Type\*);

12 [VB] Public Property Type As Type

13 [JScript] public function get Type() : Type;public function set Type(Type);

14  
15 *Description*

16       Gets or sets the type of the derived class that should be used when  
17 serializing or deserializing an object.

18       TypeId

19       SoapReflectionImporter class (System.Xml.Serialization)

20       ToString

21  
22  
23 *Description*

24  
25       SoapReflectionImporter



*Example Syntax:*

ToString

[C#] public SoapReflectionImporter();

[C++] public: SoapReflectionImporter();

[VB] Public Sub New()

[JScript] public function SoapReflectionImporter();

*Description*

SoapReflectionImporter

*Example Syntax:*

ToString

[C#] public SoapReflectionImporter(SoapAttributeOverrides attributeOverrides);

[C++] public: SoapReflectionImporter(SoapAttributeOverrides\*  
attributeOverrides);

[VB] Public Sub New(ByVal attributeOverrides As SoapAttributeOverrides)

[JScript] public function SoapReflectionImporter(attributeOverrides :  
SoapAttributeOverrides);

*Description*

SoapReflectionImporter

*Example Syntax:*

ToString

[C#] public SoapReflectionImporter(string defaultNamespace);

[C++] public: SoapReflectionImporter(String\* defaultNamespace);

[VB] Public Sub New(ByVal defaultNamespace As String)

[JScript] public function SoapReflectionImporter(defaultNamespace : String);

*Description*

SoapReflectionImporter

*Example Syntax:*

ToString

[C#] public SoapReflectionImporter(SoapAttributeOverrides attributeOverrides,  
string defaultNamespace);

[C++] public: SoapReflectionImporter(SoapAttributeOverrides\*  
attributeOverrides, String\* defaultNamespace);

[VB] Public Sub New(ByVal attributeOverrides As SoapAttributeOverrides,  
ByVal defaultNamespace As String)

[JScript] public function SoapReflectionImporter(attributeOverrides :  
SoapAttributeOverrides, defaultNamespace : String);

*Description*

ImportMembersMapping

```

1
2 [C#] public XmlMembersMapping ImportMembersMapping(string elementName,
3 string ns, XmlReflectionMember[] members);
4 [C++] public: XmlMembersMapping* ImportMembersMapping(String*
5 elementName, String* ns, XmlReflectionMember* members[]);
6 [VB] Public Function ImportMembersMapping(ByVal elementName As String,
7 ByVal ns As String, ByVal members() As XmlReflectionMember) As
8 XmlMembersMapping
9 [JScript] public function ImportMembersMapping(elementName : String, ns :
10 String, members : XmlReflectionMember[]) : XmlMembersMapping;
11

```

## *Description*

### ImportMembersMapping

```

15
16 [C#] public XmlMembersMapping ImportMembersMapping(string elementName,
17 string ns, XmlReflectionMember[] members, bool hasWrapperElement, bool
18 writeAccessors);
19 [C++] public: XmlMembersMapping* ImportMembersMapping(String*
20 elementName, String* ns, XmlReflectionMember* members[], bool
21 hasWrapperElement, bool writeAccessors);
22 [VB] Public Function ImportMembersMapping(ByVal elementName As String,
23 ByVal ns As String, ByVal members() As XmlReflectionMember, ByVal
24 hasWrapperElement As Boolean, ByVal writeAccessors As Boolean) As
25 XmlMembersMapping

```

```

1 [JScript] public function ImportMembersMapping(elementName : String, ns :
2 String, members : XmlReflectionMember[], hasWrapperElement : Boolean,
3 writeAccessors : Boolean) : XmlMembersMapping;

```

#### *Description*

#### **ImportMembersMapping**

```

9 [C#] public XmlMembersMapping ImportMembersMapping(string elementName,
10 string ns, XmlReflectionMember[] members, bool hasWrapperElement, bool
11 writeAccessors, bool validate);

```

```

12 [C++] public: XmlMembersMapping* ImportMembersMapping(String*
13 elementName, String* ns, XmlReflectionMember* members[], bool
14 hasWrapperElement, bool writeAccessors, bool validate);

```

```

15 [VB] Public Function ImportMembersMapping(ByVal elementName As String,
16 ByVal ns As String, ByVal members() As XmlReflectionMember, ByVal
17 hasWrapperElement As Boolean, ByVal writeAccessors As Boolean, ByVal
18 validate As Boolean) As XmlMembersMapping

```

```

19 [JScript] public function ImportMembersMapping(elementName : String, ns :
20 String, members : XmlReflectionMember[], hasWrapperElement : Boolean,
21 writeAccessors : Boolean, validate : Boolean) : XmlMembersMapping;

```

#### **ImportTypeMapping**

```

24 [C#] public XmlTypeMapping ImportTypeMapping(Type type);

```

```

25 [C++] public: XmlTypeMapping* ImportTypeMapping(Type* type);

```

1 [VB] Public Function ImportTypeMapping(ByVal type As Type) As

2 XmlTypeMapping

3 [JScript] public function ImportTypeMapping(type : Type) : XmlTypeMapping;

4  
5 *Description*

6  
7 ImportTypeMapping

8  
9 [C#] public XmlTypeMapping ImportTypeMapping(Type type, string  
10 defaultNamespace);

11 [C++] public: XmlTypeMapping\* ImportTypeMapping(Type\* type, String\*  
12 defaultNamespace);

13 [VB] Public Function ImportTypeMapping(ByVal type As Type, ByVal  
14 defaultNamespace As String) As XmlTypeMapping

15 [JScript] public function ImportTypeMapping(type : Type, defaultNamespace :  
16 String) : XmlTypeMapping;

17  
18 *Description*

19  
20 IncludeType

21  
22 [C#] public void IncludeType(Type type);

23 [C++] public: void IncludeType(Type\* type);

24 [VB] Public Sub IncludeType(ByVal type As Type)

25 [JScript] public function IncludeType(type : Type);

1  
2 *Description*

3  
4 IncludeTypes

5  
6 [C#] public void IncludeTypes(ICustomAttributeProvider provider);  
7 [C++] public: void IncludeTypes(ICustomAttributeProvider\* provider);  
8 [VB] Public Sub IncludeTypes(ByVal provider As ICustomAttributeProvider)  
9 [JScript] public function IncludeTypes(provider : ICustomAttributeProvider);  
10

11 *Description*

12  
13 SoapSchemaExporter class (System.Xml.Serialization)  
14 ToString  
15

16  
17 *Description*

18  
19 SoapSchemaExporter

20 *Example Syntax:*

21 ToString  
22

23 [C#] public SoapSchemaExporter(XmlSchemas schemas);  
24 [C++] public: SoapSchemaExporter(XmlSchemas\* schemas);  
25 [VB] Public Sub New(ByVal schemas As XmlSchemas)

1 [JScript] public function SoapSchemaExporter(schemas : XmlSchemas);

2  
3 *Description*

4  
5 ExportMembersMapping

6  
7 [C#] public void ExportMembersMapping(XmlMembersMapping  
8 xmlMembersMapping);

9 [C++] public: void ExportMembersMapping(XmlMembersMapping\*  
10 xmlMembersMapping);

11 [VB] Public Sub ExportMembersMapping(ByVal xmlMembersMapping As  
12 XmlMembersMapping)

13 [JScript] public function ExportMembersMapping(xmlMembersMapping :  
14 XmlMembersMapping);

15  
16 *Description*

17  
18 ExportMembersMapping

19  
20 [C#] public void ExportMembersMapping(XmlMembersMapping  
21 xmlMembersMapping, bool exportEnclosingType);

22 [C++] public: void ExportMembersMapping(XmlMembersMapping\*  
23 xmlMembersMapping, bool exportEnclosingType);

24 [VB] Public Sub ExportMembersMapping(ByVal xmlMembersMapping As  
25 XmlMembersMapping, ByVal exportEnclosingType As Boolean)

```

1 [JScript] public function ExportMembersMapping(xmlMembersMapping :
2 XmlMembersMapping, exportEnclosingType : Boolean);
3     ExportTypeMapping
4
5 [C#] public void ExportTypeMapping(XmlTypeMapping xmlTypeMapping);
6 [C++] public: void ExportTypeMapping(XmlTypeMapping* xmlTypeMapping);
7 [VB] Public Sub ExportTypeMapping(ByVal xmlTypeMapping As
8 XmlTypeMapping)
9 [JScript] public function ExportTypeMapping(xmlTypeMapping :
10 XmlTypeMapping);
11     SoapSchemaImporter class (System.Xml.Serialization)
12     ToString
13
14
15 Description
16
17     SoapSchemaImporter
18     Example Syntax:
19     ToString
20
21 [C#] public SoapSchemaImporter(XmlSchemas schemas);
22 [C++] public: SoapSchemaImporter(XmlSchemas* schemas);
23 [VB] Public Sub New(ByVal schemas As XmlSchemas)
24 [JScript] public function SoapSchemaImporter(schemas : XmlSchemas);
25

```



1  
2 *Description*

3  
4 SoapSchemaImporter

5 *Example Syntax:*

6 ToString

7  
8 [C#] public SoapSchemaImporter(XmlSchemas schemas, CodeIdentifiers  
9 typeIdentifiers);

10 [C++] public: SoapSchemaImporter(XmlSchemas\* schemas, CodeIdentifiers\*  
11 typeIdentifiers);

12 [VB] Public Sub New(ByVal schemas As XmlSchemas, ByVal typeIdentifiers As  
13 CodeIdentifiers)

14 [JScript] public function SoapSchemaImporter(schemas : XmlSchemas,  
15 typeIdentifiers : CodeIdentifiers);

16  
17 *Description*

18  
19 ImportDerivedTypeMapping

20  
21 [C#] public XmlTypeMapping ImportDerivedTypeMapping(XmlQualifiedName  
22 name, Type baseType, bool baseTypeCanBeIndirect);

23 [C++] public: XmlTypeMapping\*  
24 ImportDerivedTypeMapping(XmlQualifiedName\* name, Type\* baseType, bool  
25 baseTypeCanBeIndirect);

```

1 [VB] Public Function ImportDerivedTypeMapping(ByVal name As
2 XmlQualifiedName, ByVal baseType As Type, ByVal baseTypeCanBeIndirect As
3 Boolean) As XmlTypeMapping
4 [JScript] public function ImportDerivedTypeMapping(name : XmlQualifiedName,
5 baseType : Type, baseTypeCanBeIndirect : Boolean) : XmlTypeMapping;

```

#### *Description*

#### ImportMembersMapping

```

11 [C#] public XmlMembersMapping ImportMembersMapping(string name, string
12 ns, SoapSchemaMember member);
13 [C++] public: XmlMembersMapping* ImportMembersMapping(String* name,
14 String* ns, SoapSchemaMember* member);
15 [VB] Public Function ImportMembersMapping(ByVal name As String, ByVal ns
16 As String, ByVal member As SoapSchemaMember) As XmlMembersMapping
17 [JScript] public function ImportMembersMapping(name : String, ns : String,
18 member : SoapSchemaMember) : XmlMembersMapping;

```

#### *Description*

#### ImportMembersMapping

```

24 [C#] public XmlMembersMapping ImportMembersMapping(string name, string
25 ns, SoapSchemaMember[] members);

```

```

1 [C++] public: XmlMembersMapping* ImportMembersMapping(String* name,
2 String* ns, SoapSchemaMember* members[]);
3 [VB] Public Function ImportMembersMapping(ByVal name As String, ByVal ns
4 As String, ByVal members() As SoapSchemaMember) As XmlMembersMapping
5 [JScript] public function ImportMembersMapping(name : String, ns : String,
6 members : SoapSchemaMember[]) : XmlMembersMapping;

```

### *Description*

#### ImportMembersMapping

```

12 [C#] public XmlMembersMapping ImportMembersMapping(string name, string
13 ns, SoapSchemaMember[] members, bool hasWrapperElement);
14 [C++] public: XmlMembersMapping* ImportMembersMapping(String* name,
15 String* ns, SoapSchemaMember* members[], bool hasWrapperElement);
16 [VB] Public Function ImportMembersMapping(ByVal name As String, ByVal ns
17 As String, ByVal members() As SoapSchemaMember, ByVal hasWrapperElement
18 As Boolean) As XmlMembersMapping
19 [JScript] public function ImportMembersMapping(name : String, ns : String,
20 members : SoapSchemaMember[], hasWrapperElement : Boolean) :
21 XmlMembersMapping;

```

### *Description*

#### ImportMembersMapping

```

1  [C#] public XmlMembersMapping ImportMembersMapping(string name, string
2  ns, SoapSchemaMember[] members, bool hasWrapperElement, Type baseType,
3  bool baseTypeCanBeIndirect);
4
5  [C++] public: XmlMembersMapping* ImportMembersMapping(String* name,
6  String* ns, SoapSchemaMember* members[], bool hasWrapperElement, Type*
7  baseType, bool baseTypeCanBeIndirect);
8
9  [VB] Public Function ImportMembersMapping(ByVal name As String, ByVal ns
10 As String, ByVal members() As SoapSchemaMember, ByVal hasWrapperElement
11 As Boolean, ByVal baseType As Type, ByVal baseTypeCanBeIndirect As
12 Boolean) As XmlMembersMapping
13
14 [JScript] public function ImportMembersMapping(name : String, ns : String,
15 members : SoapSchemaMember[], hasWrapperElement : Boolean, baseType :
16 Type, baseTypeCanBeIndirect : Boolean) : XmlMembersMapping;
17
18     SoapSchemaMember class (System.Xml.Serialization)
19     ToString
20
21
22
23
24
25

```

## Description

SoapSchemaMember

*Example Syntax:*

ToString

```

[C#] public SoapSchemaMember();

```

```

1  [C++] public: SoapSchemaMember();
2  [VB] Public Sub New()
3  [JScript] public function SoapSchemaMember();
4      MemberName
5      ToString
6
7  [C#] public string MemberName {get; set;}
8  [C++] public: __property String* get_MemberName();public: __property void
9  set_MemberName(String*);
10 [VB] Public Property MemberName As String
11 [JScript] public function get MemberName() : String;public function set
12 MemberName(String);
13
14 Description
15
16      MemberType
17      ToString
18
19 [C#] public XmlQualifiedName MemberType {get; set;}
20 [C++] public: __property XmlQualifiedName* get_MemberType();public:
21 __property void set_MemberType(XmlQualifiedName*);
22 [VB] Public Property MemberType As XmlQualifiedName
23 [JScript] public function get MemberType() : XmlQualifiedName;public function
24 set MemberType(XmlQualifiedName);
25

```

*Description*

SoapTypeAttribute class (System.Xml.Serialization)

ToString

*Description*

Specifies that the class should be serialized by the XmlSerializer as a complex type XML element.

The **System.Xml.Serialization.SoapTypeAttribute** belongs to a family of attributes that controls how the **System.Xml.Serialization.XmlSerializer** serializes or deserializes an object. For a complete list of similar attributes, see the **System.Xml.Serialization.XmlSerializer** class.

SoapTypeAttribute

*Example Syntax:*

ToString

[C#] public SoapTypeAttribute();

[C++] public: SoapTypeAttribute();

[VB] Public Sub New()

[JScript] public function SoapTypeAttribute(); Initializes a new instance of the **System.Xml.Serialization.SoapTypeAttribute** class.

*Description*

1        Initializes a new instance of the  
2 **System.Xml.Serialization.SoapTypeAttribute** class.

3        SoapTypeAttribute

4        *Example Syntax:*

5        ToString

7 [C#] public SoapTypeAttribute(string typeName);

8 [C++] public: SoapTypeAttribute(String\* typeName);

9 [VB] Public Sub New(ByVal typeName As String)

10 [JScript] public function SoapTypeAttribute(typeName : String);

12 *Description*

13        Initializes a new instance of the  
14 **System.Xml.Serialization.SoapTypeAttribute** class specifying the name of the  
15 XML type.

16        SoapTypeAttribute

17        *Example Syntax:*

18        ToString

20 [C#] public SoapTypeAttribute(string typeName, string ns);

21 [C++] public: SoapTypeAttribute(String\* typeName, String\* ns);

22 [VB] Public Sub New(ByVal typeName As String, ByVal ns As String)

23 [JScript] public function SoapTypeAttribute(typeName : String, ns : String);

25 *Description*

1        Initializes a new instance of the  
2 **System.Xml.Serialization.SoapTypeAttribute** class specifying the name, and  
3 XML namespace, of the type.

4        IncludeInSchema

5        ToString

6  
7 [C#] public bool IncludeInSchema {get; set;}

8 [C++] public: \_\_property bool get\_IncludeInSchema();public: \_\_property void  
9 set\_IncludeInSchema(bool);

10 [VB] Public Property IncludeInSchema As Boolean

11 [JScript] public function get IncludeInSchema() : Boolean;public function set  
12 IncludeInSchema(Boolean);

13  
14 *Description*

15        Gets or sets a value indicating whether to include the type in SOAP-  
16 encoded XML schema documents.

17        Apply the **System.Xml.Serialization.SoapTypeAttribute** to a class to  
18 specify the XML type's namespace, the XML type name, and whether to include  
19 the type in the XML schema document. To see the results of setting the  
20 **System.Xml.Serialization.XmlTypeAttribute** class's properties, compile your  
21 application as an executable or DLL, and pass the resulting file to the XML  
22 Schema Definition tool (XSD.exe). The tool writes the schema--including the type  
23 definition.

24        Namespace

25        ToString



```

1  [C#] public string Namespace {get; set;}
2
3  [C++] public: __property String* get_Namespace();public: __property void
4  set_Namespace(String*);
5
6  [VB] Public Property Namespace As String
7
8  [JScript] public function get Namespace() : String;public function set
9  Namespace(String);

```

#### *Description*

Gets or sets the namespace of the XML type.

TypeId

TypeName

ToString

#### *Description*

Gets or sets the name of the XML type.

UnreferencedObjectEventArgs class (System.Xml.Serialization)

ToString

UnreferencedObjectEventArgs

#### *Example Syntax:*

ToString

UnreferencedId

ToString

UnreferencedObject

ToString

UnreferencedObjectEventHandler delegate (System.Xml.Serialization)

ToString

XmlAnyAttributeAttribute class (System.Xml.Serialization)

ToString

### *Description*

Specifies that the member, usually a field that returns an array, captures all XML attributes found in a XML document.

The **System.Xml.Serialization.XmlAnyAttributeAttribute** can be applied to a field, parameter, property, or return value of type **System.Xml.XmlElement** or **System.Xml.XmlNode**.

XmlAnyAttributeAttribute

### *Example Syntax:*

ToString

[C#] public XmlAnyAttributeAttribute();

[C++] public: XmlAnyAttributeAttribute();

[VB] Public Sub New()

[JScript] public function XmlAnyAttributeAttribute(); Constructs a new instance of the **System.Xml.Serialization.XmlAnyAttributeAttribute** class.

### *Description*

Constructs a new instance of the  
**System.Xml.Serialization.XmlAnyAttributeAttribute** class.

**XmlAnyAttributeAttribute**

*Example Syntax:*

**ToString**

[C#] public XmlAnyAttributeAttribute(string name);

[C++] public: XmlAnyAttributeAttribute(String\* name);

[VB] Public Sub New(ByVal name As String)

[JScript] public function XmlAnyAttributeAttribute(name : String);

#### *Description*

Constructs a new instance of the  
**System.Xml.Serialization.XmlAnyAttributeAttribute** class using the specified  
name. The name of the XML attribute.

**XmlAnyAttributeAttribute**

*Example Syntax:*

**ToString**

[C#] public XmlAnyAttributeAttribute(string name, string ns);

[C++] public: XmlAnyAttributeAttribute(String\* name, String\* ns);

[VB] Public Sub New(ByVal name As String, ByVal ns As String)

[JScript] public function XmlAnyAttributeAttribute(name : String, ns : String);

#### *Description*

Constructs a new instance of the **System.Xml.Serialization.XmlAnyAttributeAttribute** class; specifies the name and namespace to be generated in the XML document. The name of the XML attribute. The XML namespace of the XML attribute.

Name

ToString

[C#] public string Name {get; set;}

[C++] public: \_\_property String\* get\_Name();public: \_\_property void set\_Name(String\*);

[VB] Public Property Name As String

[JScript] public function get Name() : String;public function set Name(String);

#### *Description*

Gets or sets the name of the XML attribute generated in the XML document.

Namespace

ToString

[C#] public string Namespace {get; set;}

[C++] public: \_\_property String\* get\_Namespace();public: \_\_property void set\_Namespace(String\*);

[VB] Public Property Namespace As String

[JScript] public function get Namespace() : String;public function set Namespace(String);

1  
2 *Description*

3 Gets or sets the XML namespace of the XML attribute generated in the  
4 XML document.

5 `TypeId`

6 `XmlAnyElementAttribute` class (System.Xml.Serialization)

7 `ToString`  
8  
9

10 *Description*

11 Specifies that the member, usually a field that returns an array, captures all  
12 XML elements found in a XML document being deserialized.

13 Apply the **System.Xml.Serialization.XmlAnyElementAttribute** to a field  
14 that returns an array of **System.Xml.XmlElement** objects.

15 `XmlAnyElementAttribute`

16 *Example Syntax:*

17 `ToString`  
18

19 `[C#] public XmlAnyElementAttribute();`

20 `[C++] public: XmlAnyElementAttribute();`

21 `[VB] Public Sub New()`

22 `[JScript] public function XmlAnyElementAttribute();` Initializes a new instance of  
23 the **System.Xml.Serialization.XmlAnyElementAttribute** class.  
24

25 *Description*

1        Initializes a new instance of the  
2 **System.Xml.Serialization.XmlAnyElementAttribute** class.

3        XmlAnyElementAttribute

4        *Example Syntax:*

5        ToString

6  
7 [C#] public XmlAnyElementAttribute(string name);

8 [C++] public: XmlAnyElementAttribute(String\* name);

9 [VB] Public Sub New(ByVal name As String)

10 [JScript] public function XmlAnyElementAttribute(name : String);

11  
12 *Description*

13        Initializes a new instance of the  
14 **System.Xml.Serialization.XmlAnyElementAttribute** class; specifies the XML  
15 element name generated in the XML-document instance. The name of the XML  
16 element that the **System.Xml.Serialization.XmlSerializer** generates.

17        XmlAnyElementAttribute

18        *Example Syntax:*

19        ToString

20  
21 [C#] public XmlAnyElementAttribute(string name, string ns);

22 [C++] public: XmlAnyElementAttribute(String\* name, String\* ns);

23 [VB] Public Sub New(ByVal name As String, ByVal ns As String)

24 [JScript] public function XmlAnyElementAttribute(name : String, ns : String);

## Description

Initializes a new instance of the **System.Xml.Serialization.XmlAnyElementAttribute** class; specifies the XML element name generated in the XML-document instance and its XML namespace. The name of the XML element that the **System.Xml.Serialization.XmlSerializer** generates. The XML namespace of the XML element.

Name

ToString

```
[C#] public string Name {get; set;}
```

```
[C++] public: __property String* get_Name();public: __property void  
set_Name(String*);
```

```
[VB] Public Property Name As String
```

```
[JScript] public function get Name() : String;public function set Name(String);
```

## Description

Gets or sets the XML element name.

Namespace

ToString

```
[C#] public string Namespace {get; set;}
```

```
[C++] public: __property String* get_Namespace();public: __property void  
set_Namespace(String*);
```

```
[VB] Public Property Namespace As String
```

[JScript] public function get Namespace() : String; public function set  
Namespace(String);

#### *Description*

Gets or sets the XML namespace generated in the XML document.

TypeId

XmlAnyElementAttributes class (System.Xml.Serialization)

ToString

#### *Description*

Represents a collection of

**System.Xml.Serialization.XmlAnyElementAttribute** objects.

Use the **System.Xml.Serialization.XmlAnyElementAttributes** to  
override the behavior of a set of

**System.Xml.Serialization.XmlAnyElementAttribute** objects.

XmlAnyElementAttributes

*Example Syntax:*

ToString

[C#] public XmlAnyElementAttributes();

[C++] public: XmlAnyElementAttributes();

[VB] Public Sub New()

[JScript] public function XmlAnyElementAttributes();

Count



InnerList

Item

ToString

### Description

Gets or sets the **System.Xml.Serialization.XmlAnyElementAttribute** at the specified index. The index of the **System.Xml.Serialization.XmlAnyElementAttribute**.

List

Add

[C#] public int Add(XmlAnyElementAttribute attribute);

[C++] public: int Add(XmlAnyElementAttribute\* attribute);

[VB] Public Function Add(ByVal attribute As XmlAnyElementAttribute) As

Integer

[JScript] public function Add(attribute : XmlAnyElementAttribute) : int;

### Description

Adds an **System.Xml.Serialization.XmlAnyElementAttribute** to the collection.

*Return Value:* The index of the newly added

**System.Xml.Serialization.XmlAnyElementAttribute** . The

**System.Xml.Serialization.XmlAnyElementAttribute** to add.

Contains

```

1
2 [C#] public bool Contains(XmlAnyElementAttribute attribute);
3 [C++] public: bool Contains(XmlAnyElementAttribute* attribute);
4 [VB] Public Function Contains(ByVal attribute As XmlAnyElementAttribute) As
5 Boolean
6 [JScript] public function Contains(attribute : XmlAnyElementAttribute) : Boolean;
7

```

### *Description*

Gets a value indicating whether the specified **System.Xml.Serialization.XmlAnyElementAttribute** exists in the collection.

*Return Value:* **true** if the **System.Xml.Serialization.XmlAnyElementAttribute** exists in the collection; otherwise, **false** . The **System.Xml.Serialization.XmlAnyElementAttribute** you are interested in.

### *CopyTo*

```

14
15
16 [C#] public void CopyTo(XmlAnyElementAttribute[] array, int index);
17 [C++] public: void CopyTo(XmlAnyElementAttribute* array[], int index);
18 [VB] Public Sub CopyTo(ByVal array() As XmlAnyElementAttribute, ByVal
19 index As Integer)
20 [JScript] public function CopyTo(array : XmlAnyElementAttribute[], index : int);
21

```

### *Description*

### *IndexOf*

```

1
2 [C#] public int IndexOf(XmlAnyElementAttribute attribute);
3 [C++] public: int IndexOf(XmlAnyElementAttribute* attribute);
4 [VB] Public Function IndexOf(ByVal attribute As XmlAnyElementAttribute) As
5 Integer
6 [JScript] public function IndexOf(attribute : XmlAnyElementAttribute) : int;
7

```

### *Description*

Gets the index of the specified

**System.Xml.Serialization.XmlAnyElementAttribute** .

*Return Value:* The index of the specified

**System.Xml.Serialization.XmlAnyElementAttribute** . The

**System.Xml.Serialization.XmlAnyElementAttribute** whose index you want.

### *Insert*

```

14
15
16 [C#] public void Insert(int index, XmlAnyElementAttribute attribute);
17 [C++] public: void Insert(int index, XmlAnyElementAttribute* attribute);
18 [VB] Public Sub Insert(ByVal index As Integer, ByVal attribute As
19 XmlAnyElementAttribute)
20 [JScript] public function Insert(index : int, attribute : XmlAnyElementAttribute);
21

```

### *Description*

Inserts an **System.Xml.Serialization.XmlAnyElementAttribute** into the collection at the specified index. The index where the

**System.Xml.Serialization.XmlAnyElementAttribute** will be inserted. The  
**System.Xml.Serialization.XmlAnyElementAttribute** to insert.

Remove

[C#] public void Remove(XmlAnyElementAttribute attribute);

[C++] public: void Remove(XmlAnyElementAttribute\* attribute);

[VB] Public Sub Remove(ByVal attribute As XmlAnyElementAttribute)

[JScript] public function Remove(attribute : XmlAnyElementAttribute);

### *Description*

Removes the specified

**System.Xml.Serialization.XmlAnyElementAttribute** from the collection. The  
**System.Xml.Serialization.XmlAnyElementAttribute** to remove.

XmlAttribute class (System.Xml.Serialization)

ToString

### *Description*

Specifies that the **System.Xml.Serialization.XmlSerializer** should  
serialize a particular class member as an array of XML elements.

The **System.Xml.Serialization.XmlArrayAttribute** belongs to a family of  
attributes that controls how the **System.Xml.Serialization.XmlSerializer**  
serializes, or deserializes, an object. For a complete list of similar attributes, see  
the **System.Xml.Serialization.XmlSerializer** class.

XmlAttribute

*Example Syntax:*

ToString

[C#] public XmlArrayAttribute();

[C++] public: XmlArrayAttribute();

[VB] Public Sub New()

[JScript] public function XmlArrayAttribute(); Initializes a new instance of the **System.Xml.Serialization.XmlArrayAttribute** class.

### *Description*

Initializes a new instance of the **System.Xml.Serialization.XmlArrayAttribute** class.

For more information about using attributes, see .

XmlArrayAttribute

*Example Syntax:*

ToString

[C#] public XmlArrayAttribute(string elementName);

[C++] public: XmlArrayAttribute(String\* elementName);

[VB] Public Sub New(ByVal elementName As String)

[JScript] public function XmlArrayAttribute(elementName : String);

### *Description*

1        Initializes a new instance of the  
2        **System.Xml.Serialization.XmlArrayAttribute** class; specifies the XML element  
3        name generated in the XML-document instance.

4        For more information about using attributes, see . The name of the XML  
5        element that the **System.Xml.Serialization.XmlSerializer** generates.

6        ElementName

7        ToString

9        [C#] public string ElementName {get; set;}

10       [C++] public: \_\_property String\* get\_ElementName();public: \_\_property void  
11       set\_ElementName(String\*);

12       [VB] Public Property ElementName As String

13       [JScript] public function get ElementName() : String;public function set  
14       ElementName(String);

16       *Description*

17       Gets or sets the XML element name given to the serialized array.

18       Specify an **System.Xml.Serialization.XmlArrayAttribute.ElementName**  
19       when you want the generated XML element name to differ from the member's  
20       identifier.

21       Form

22       ToString

24       [C#] public XmlSchemaForm Form {get; set;}

25       [C++] public: \_\_property XmlSchemaForm get\_Form();public: \_\_property void

1 set\_Form(XmlSchemaForm);

2 [VB] Public Property Form As XmlSchemaForm

3 [JScript] public function get Form() : XmlSchemaForm;public function set

4 Form(XmlSchemaForm);

5  
6 *Description*

7 Gets or sets a value indicating whether the XML element name generated  
8 by the **System.Xml.Serialization.XmlSerializer** is qualified or unqualified.

9 The **System.Xml.Serialization.XmlAttributeAttribute.Form** property  
10 determines whether an XML element name is qualified or unqualified. The  
11 **System.Xml.Serialization.XmlAttributeAttribute.Form** property conforms to  
12 the 1999 <http://www.w3.org> specification *Namespaces in XML* .

13       IsNullable

14       ToString

15  
16 [C#] public bool IsNullable {get; set;}

17 [C++] public: \_\_property bool get\_IsNullable();public: \_\_property void

18 set\_IsNullable(bool);

19 [VB] Public Property IsNullable As Boolean

20 [JScript] public function get IsNullable() : Boolean;public function set

21 IsNullable(Boolean);

22  
23 *Description*

1 Gets or sets a value indicating whether the  
2 **System.Xml.Serialization.XmlSerializer** should serialize a member that is set to  
3 **null** into the **xsi:nil** attribute set to **true** .

4 The XML schema specification for structures allows an XML document to  
5 explicitly signal that an element's content is missing. Such an element contains the  
6 attribute **xsi:nil** set to **true** . For more information, see the  
7 <http://www.w3.org/TR/xmlschema-1/> specification named *XML Schema Part 1:*  
8 *Structures* .

9 Namespace

10 ToString

11  
12 [C#] public string Namespace {get; set;}

13 [C++] public: \_\_property String\* get\_Namespace();public: \_\_property void  
14 set\_Namespace(String\*);

15 [VB] Public Property Namespace As String

16 [JScript] public function get Namespace() : String;public function set  
17 Namespace(String);

### 18 19 *Description*

20 Gets or set the namespace of the XML element.

21 The **System.Xml.Serialization.XmlArrayAttribute.Namespace** property  
22 allows you to create qualified XML element names. The  
23 **System.Xml.Serialization.XmlArrayAttribute.Namespace** property conforms  
24 to the rules for creating an XML namespace as found in the <http://www.w3.org>  
25 document *Namespaces in XML* .



1       TypeId

2       XmlArrayItemAttribute class (System.Xml.Serialization)

3       ToString

6       *Description*

7           Specifies the derived types that the  
8       **System.Xml.Serialization.XmlSerializer** can place in a serialized array.

9           The **System.Xml.Serialization.XmlArrayItemAttribute** belongs to a  
10       family of attributes that controls how the  
11       **System.Xml.Serialization.XmlSerializer** serializes, or deserializes, an object. For  
12       a complete list of similar attributes, see the  
13       **System.Xml.Serialization.XmlSerializer** class.

14       XmlArrayItemAttribute

15       *Example Syntax:*

16       ToString

18       [C#] public XmlArrayItemAttribute();

19       [C++] public: XmlArrayItemAttribute();

20       [VB] Public Sub New()

21       [JScript] public function XmlArrayItemAttribute(); Initializes a new instance of  
22       the **System.Xml.Serialization.XmlArrayItemAttribute** class.

24       *Description*

1        Initializes a new instance of the  
2 **System.Xml.Serialization.XmlArrayItemAttribute** class.

3        XmlArrayItemAttribute

4        *Example Syntax:*

5        ToString

7 [C#] public XmlArrayItemAttribute(string elementName);

8 [C++] public: XmlArrayItemAttribute(String\* elementName);

9 [VB] Public Sub New(ByVal elementName As String)

10 [JScript] public function XmlArrayItemAttribute(elementName : String);

## 12 *Description*

13        Initializes a new instance of the  
14 **System.Xml.Serialization.XmlArrayItemAttribute** class; specifies the name of  
15 the XML element generated in the XML-document instance.

16        This overload sets the  
17 **System.Xml.Serialization.XmlArrayItemAttribute.ElementName** property.

18        The name of the XML element.

19        XmlArrayItemAttribute

20        *Example Syntax:*

21        ToString

23 [C#] public XmlArrayItemAttribute(Type type);

24 [C++] public: XmlArrayItemAttribute(Type\* type);

25 [VB] Public Sub New(ByVal type As Type)

[JScript] public function XmlArrayItemAttribute(type : Type);

*Description*

Initializes a new instance of the **System.Xml.Serialization.XmlArrayItemAttribute** class; specifies the **System.Type** that can be inserted into the serialized array. The **System.Type** of the object to serialize.

**XmlArrayItemAttribute**

*Example Syntax:*

**ToString**

[C#] public XmlArrayItemAttribute(string elementName, Type type);

[C++] public: XmlArrayItemAttribute(String\* elementName, Type\* type);

[VB] Public Sub New(ByVal elementName As String, ByVal type As Type)

[JScript] public function XmlArrayItemAttribute(elementName : String, type : Type);

*Description*

Initializes a new instance of the **System.Xml.Serialization.XmlArrayItemAttribute** class; specifies the name of the XML element generated in the XML-document instance, and the **System.Type** that can be inserted into the generated XML-document instance.

This overload sets the **System.Xml.Serialization.XmlArrayItemAttribute.ElementName** and the

**System.Xml.Serialization.XmlArrayItemAttribute.Type** properties. The name of the XML element. The **System.Type** of the object to serialize.

DataType

ToString

[C#] public string DataType {get; set;}

[C++] public: \_\_property String\* get\_DataType();public: \_\_property void set\_DataType(String\*);

[VB] Public Property DataType As String

[JScript] public function get DataType() : String;public function set DataType(String);

### *Description*

Gets or sets the XML data type of the generated XML element.

The following table lists the XML data types with their .NET equivalents.

For the XML **date** and **time** data types, the .NET data type consists of a **System.DateTime** to which the

**System.Xml.Serialization.XmlElementAttribute** has been applied with the

**System.Xml.Serialization.XmlElementAttribute.DataType** set to "date" and "time", respectively.

ElementName

ToString

[C#] public string ElementName {get; set;}

[C++] public: \_\_property String\* get\_ElementName();public: \_\_property void

1 set\_ElementName(String\*);

2 [VB] Public Property ElementName As String

3 [JScript] public function get ElementName() : String;public function set

4 ElementName(String);

6 *Description*

7 Gets or sets the name of the generated XML element.

8 Specify an

9 **System.Xml.Serialization.XmlArrayItemAttribute.ElementName** if you want  
10 the name of the generated XML element to differ from the member's identifier.

11 Form

12 ToString

14 [C#] public XmlSchemaForm Form {get; set;}

15 [C++] public: \_\_property XmlSchemaForm get\_Form();public: \_\_property void

16 set\_Form(XmlSchemaForm);

17 [VB] Public Property Form As XmlSchemaForm

18 [JScript] public function get Form() : XmlSchemaForm;public function set

19 Form(XmlSchemaForm);

21 *Description*

22 Gets or sets a value indicating whether the name of the generated XML  
23 element is qualified.

The **System.Xml.Serialization.XmlAttributeAttribute.Form** property determines whether an XML element name is qualified, based on the <http://www.w3.org> specification *Namespaces in XML* .

IsNullable

ToString

[C#] public bool IsNullable {get; set;}

[C++] public: \_\_property bool get\_IsNullable();public: \_\_property void set\_IsNullable(bool);

[VB] Public Property IsNullable As Boolean

[JScript] public function get IsNullable() : Boolean;public function set IsNullable(Boolean);

#### *Description*

Gets or sets a value indicating whether the **System.Xml.Serialization.XmlSerializer** should serialize a member that is set to **null** into the **xsi:nil** attribute set to **true** .

The XML schema specification for structures allows an XML document to explicitly signal that an element's content is missing. Such an element contains the attribute **xsi:nil** set to **true** . For more information, see the <http://www.w3.org/TR/xmlschema-1/> specification named *XML Schema Part 1: Structures* .

Namespace

ToString

[C#] public string Namespace {get; set;}

[C++] public: \_\_property String\* get\_Namespace();public: \_\_property void

set\_Namespace(String\*);

[VB] Public Property Namespace As String

[JScript] public function get Namespace() : String;public function set

Namespace(String);

### *Description*

Gets or sets the namespace of the generated XML element.

The **System.Xml.Serialization.XmlArrayItemAttribute.Namespace** property conforms to the <http://www.w3.org> specification *Namespaces in XML* .

NestingLevel

ToString

[C#] public int NestingLevel {get; set;}

[C++] public: \_\_property int get\_NestingLevel();public: \_\_property void

set\_NestingLevel(int);

[VB] Public Property NestingLevel As Integer

[JScript] public function get NestingLevel() : int;public function set

NestingLevel(int);

### *Description*

Gets or sets the level in a hierarchy of XML elements that the

**System.Xml.Serialization.XmlArrayItemAttribute** affects.

An XML document can contain hierarchies of XML elements. To represent such a hierarchy, an array of arrays is used. In such an array, each index represents a level in the hierarchy. Therefore, the **System.Xml.Serialization.XmlArrayItemAttribute.NestingLevel** property is only used when applying an **System.Xml.Serialization.XmlArrayItemAttribute** to a field that returns an array of arrays.

Type

ToString

[C#] public Type Type {get; set;}

[C++] public: \_\_property Type\* get\_Type();public: \_\_property void set\_Type(Type\*);

[VB] Public Property Type As Type

[JScript] public function get Type() : Type;public function set Type(Type);

#### *Description*

Gets or sets the type of the elements in the generated XML array.

Use the **System.Xml.Serialization.XmlElementAttribute.Type** property to specify an overridden type for a public field or public read/write property value.

TypeId

XmlArrayItemAttributes class (System.Xml.Serialization)

ToString

#### *Description*



Represents a collection of  
**System.Xml.Serialization.XmlArrayItemAttribute** objects.

The **System.Xml.Serialization.XmlArrayItemAttributes** class allows you to specify the derived types that can be inserted into an array returned by a public field or public read/write property.

**XmlArrayItemAttributes**

*Example Syntax:*

**ToString**

[C#] public XmlArrayItemAttributes();

[C++] public: XmlArrayItemAttributes();

[VB] Public Sub New()

[JScript] public function XmlArrayItemAttributes();

**Count**

**InnerList**

**Item**

**ToString**

*Description*

Gets or sets the item at the specified index. The zero-based index of the collection member to get or set.

**List**

**Add**

```

1
2 [C#] public int Add(XmlArrayItemAttribute attribute);
3 [C++] public: int Add(XmlArrayItemAttribute* attribute);
4 [VB] Public Function Add(ByVal attribute As XmlArrayItemAttribute) As Integer
5 [JScript] public function Add(attribute : XmlArrayItemAttribute) : int;

```

#### *Description*

Adds an **System.Xml.Serialization.XmlArrayItemAttribute** to the collection.

*Return Value:* The index of the added item. The **System.Xml.Serialization.XmlArrayItemAttribute** to add to the collection.

#### Contains

```

12
13
14 [C#] public bool Contains(XmlArrayItemAttribute attribute);
15 [C++] public: bool Contains(XmlArrayItemAttribute* attribute);
16 [VB] Public Function Contains(ByVal attribute As XmlArrayItemAttribute) As
17 Boolean
18 [JScript] public function Contains(attribute : XmlArrayItemAttribute) : Boolean;

```

#### *Description*

#### CopyTo

```

22
23
24 [C#] public void CopyTo(XmlArrayItemAttribute[] array, int index);
25 [C++] public: void CopyTo(XmlArrayItemAttribute* array[], int index);

```



1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25

Remove

```
[C#] public void Remove(XmlArrayItemAttribute attribute);  
[C++] public: void Remove(XmlArrayItemAttribute* attribute);  
[VB] Public Sub Remove(ByVal attribute As XmlArrayItemAttribute)  
[JScript] public function Remove(attribute : XmlArrayItemAttribute);
```

*Description*

XmlAttributeAttribute class (System.Xml.Serialization)  
ToString

*Description*

Specifies that the **System.Xml.Serialization.XmlSerializer** should  
serialize the class member as an XML attribute.  
  
The **System.Xml.Serialization.XmlAttributeAttribute** belongs to a  
family of attributes that controls how the  
**System.Xml.Serialization.XmlSerializer** serializes, or deserializes, an object. For  
a complete list of similar attributes, see the  
**System.Xml.Serialization.XmlSerializer** class.

XmlAttributeAttribute  
*Example Syntax:*  
ToString

[C#] public XmlAttributeAttribute();  
 [C++] public: XmlAttributeAttribute();  
 [VB] Public Sub New()  
 [JScript] public function XmlAttributeAttribute(); Initializes a new instance of the **System.Xml.Serialization.XmlAttributeAttribute** class.

#### *Description*

Initializes a new instance of the **System.Xml.Serialization.XmlAttributeAttribute** class.

XmlAttributeAttribute

*Example Syntax:*

ToString

[C#] public XmlAttributeAttribute(string attributeName);  
 [C++] public: XmlAttributeAttribute(String\* attributeName);  
 [VB] Public Sub New(ByVal attributeName As String)  
 [JScript] public function XmlAttributeAttribute(attributeName : String);

#### *Description*

Initializes a new instance of the **System.Xml.Serialization.XmlAttributeAttribute** class and specifies the name of the generated XML attribute. The name of the XML attribute that the **System.Xml.Serialization.XmlSerializer** generates.

XmlAttributeAttribute

*Example Syntax:*

ToString

[C#] public XmlAttributeAttribute(Type type);

[C++] public: XmlAttributeAttribute(Type\* type);

[VB] Public Sub New(ByVal type As Type)

[JScript] public function XmlAttributeAttribute(type : Type);

*Description*

Initializes a new instance of the **System.Xml.Serialization.XmlAttributeAttribute** class. The **System.Type** used to store the attribute.

XmlAttributeAttribute

*Example Syntax:*

ToString

[C#] public XmlAttributeAttribute(string attributeName, Type type);

[C++] public: XmlAttributeAttribute(String\* attributeName, Type\* type);

[VB] Public Sub New(ByVal attributeName As String, ByVal type As Type)

[JScript] public function XmlAttributeAttribute(attributeName : String, type : Type);

*Description*

Initializes a new instance of the **System.Xml.Serialization.XmlAttributeAttribute** class. The name of the XML attribute that is generated. The **System.Type** used to store the attribute.

AttributeName

ToString

[C#] public string AttributeName {get; set;}

[C++] public: \_\_property String\* get\_AttributeName();public: \_\_property void set\_AttributeName(String\*);

[VB] Public Property AttributeName As String

[JScript] public function get AttributeName() : String;public function set AttributeName(String);

#### *Description*

Gets or sets the name of the XML attribute.

Use the

**System.Xml.Serialization.XmlAttributeAttribute.AttributeName** property to specify an XML attribute name when the default value cannot be used. For example, if the XML attribute name is illegal as a member identifier, you can use a legal name for the identifier while setting the **System.Xml.Serialization.XmlAttributeAttribute.AttributeName** to an illegal name.

DataType

ToString

[C#] public string DataType {get; set;}

[C++] public: \_\_property String\* get\_DataType();public: \_\_property void

set\_DataType(String\*);

[VB] Public Property DataType As String

[JScript] public function get DataType() : String;public function set

DataType(String);

### Description

Gets or sets the XML data type of the XML attribute generated by the **System.Xml.Serialization.XmlSerializer**.

The following table lists the XML data types with their .NET equivalents.

For the XML **date** and **time** data types, the .NET data type consists of a

**System.DateTime** to which the

**System.Xml.Serialization.XmlElementAttribute** has been applied with the

**System.Xml.Serialization.XmlElementAttribute.DataType** set to "date" and "time", respectively.

Form

ToString

[C#] public XmlSchemaForm Form {get; set;}

[C++] public: \_\_property XmlSchemaForm get\_Form();public: \_\_property void

set\_Form(XmlSchemaForm);

[VB] Public Property Form As XmlSchemaForm

[JScript] public function get Form() : XmlSchemaForm;public function set



Form(XmlSchemaForm);

### Description

Gets or sets a value indicating whether the XML attribute name generated by the **System.Xml.Serialization.XmlSerializer** is qualified.

The **System.Xml.Serialization.XmlAttributeAttribute.Form** property determines whether an XML element is qualified or unqualified. The **System.Xml.Serialization.XmlAttributeAttribute.Form** property conforms to the 1999 <http://www.w3.org> specification *Namespaces in XML*.

Namespace

ToString

[C#] public string Namespace {get; set;}

[C++] public: \_\_property String\* get\_Namespace();public: \_\_property void set\_Namespace(String\*);

[VB] Public Property Namespace As String

[JScript] public function get Namespace() : String;public function set Namespace(String);

### Description

Gets or sets the XML namespace of the XML attribute.

The **System.Xml.Serialization.XmlAttributeAttribute.Namespace** property conforms to the <http://www.w3.org> specification *Namespaces in XML*.

Type

ToString

```

1
2 [C#] public Type Type {get; set;}
3 [C++] public: __property Type* get_Type();public: __property void
4 set_Type(Type*);
5 [VB] Public Property Type As Type
6 [JScript] public function get Type() : Type;public function set Type(Type);

```

#### Description

Gets or sets the complex type of the XML attribute.

TypeId

XmlAttributeEventArgs class (System.Xml.Serialization)

ToString

#### Description

Provides data for the

**System.Xml.Serialization.XmlSerializer.UnknownAttribute** event.

For more information about handling events, see .

Attr

ToString

```

22 [C#] public XmlAttribute Attr {get;}
23 [C++] public: __property XmlAttribute* get_Attr();
24 [VB] Public ReadOnly Property Attr As XmlAttribute
25 [JScript] public function get Attr() : XmlAttribute;

```

*Description*

Gets an object that represents the unknown XML attribute.

LineNumber

ToString

[C#] public int LineNumber {get;}

[C++] public: \_\_property int get\_LineNumber();

[VB] Public ReadOnly Property LineNumber As Integer

[JScript] public function get LineNumber() : int;

*Description*

Gets the line number of the unknown XML attribute.

LinePosition

ToString

[C#] public int LinePosition {get;}

[C++] public: \_\_property int get\_LinePosition();

[VB] Public ReadOnly Property LinePosition As Integer

[JScript] public function get LinePosition() : int;

*Description*

Gets the position in the line of the unknown XML attribute.

ObjectBeingDeserialized

ToString

1  
2 [C#] public object ObjectBeingDeserialized {get;}

3 [C++] public: \_\_property Object\* get\_ObjectBeingDeserialized();

4 [VB] Public ReadOnly Property ObjectBeingDeserialized As Object

5 [JScript] public function get ObjectBeingDeserialized() : Object;

6  
7 *Description*

8 Gets the object being deserialized.

9 XmlAttributeEventHandler delegate (System.Xml.Serialization)

10 ToString

11  
12  
13 *Description*

14 Represents the method that will handle the

15 **System.Xml.Serialization.XmlSerializer.UnknownAttribute** The source of the  
16 event. An **System.Xml.Serialization.XmlAttributeEventArgs** that contains the  
17 event data.

18 When you create an

19 **System.Xml.Serialization.XmlAttributeEventHandler** delegate, you identify  
20 the method that will handle the event. To associate the event with your event  
21 handler, add an instance of the delegate to the event. The event handler is called  
22 whenever the event occurs, unless you remove the delegate. For more information  
23 about event handler delegates, see .

24 XmlAttributeOverrides class (System.Xml.Serialization)

25 ToString

### *Description*

Allows you to override property, field, and class attributes when you use the **System.Xml.Serialization.XmlSerializer** to serialize or deserialize an object.

The **System.Xml.Serialization.XmlAttributeOverrides** enables the **System.Xml.Serialization.XmlSerializer** to override the default way of serializing a set of objects. Overriding serialization in this way has two uses: first, you can control and augment the serialization of objects found in a DLL--even if you don't have access to the source; second, you can create one set of serializable classes, but serialize the objects in multiple ways. For example, instead of serializing members of a class instance as XML elements, you can serialize them as XML attributes, resulting in a more efficient document to transport.

**XmlAttributeOverrides**

*Example Syntax:*

**ToString**

```
[C#] public XmlAttributeOverrides();
```

```
[C++] public: XmlAttributeOverrides();
```

```
[VB] Public Sub New()
```

```
[JScript] public function XmlAttributeOverrides();
```

**Item**

**ToString**

```
[C#] public XmlAttributes this[Type type] {get;}
```

1 [C++] public: \_\_property XmlAttributes\* get\_Item(Type\* type);  
 2 [VB] Public Default ReadOnly Property Item(ByVal type As Type) As  
 3 XmlAttributes  
 4 [JScript] returnValue = XmlAttributeOverridesObject.Item(type); Gets an object  
 5 that represents the collection of overriding attributes.

#### 7 *Description*

8 Gets the object associated with the specified, base-class, type.

9 Use this overload to return an **System.Xml.Serialization.XmlAttributes**  
 10 object that contains attributes for an  
 11 **System.Xml.Serialization.XmlRootAttribute** or  
 12 **System.Xml.Serialization.XmlTypeAttribute** object. The base-class class  
 13 **System.Type** that is associated with the collection of attributes you want.

14 Item

15 ToString

17 [C#] public XmlAttributes this[Type type, string member] {get;}  
 18 [C++] public: \_\_property XmlAttributes\* get\_Item(Type\* type, String\* member);  
 19 [VB] Public Default ReadOnly Property Item(ByVal type As Type, ByVal  
 20 member As String) As XmlAttributes  
 21 [JScript] returnValue = XmlAttributeOverridesObject.Item(type, member);

#### 23 *Description*

24 Gets the object associated with the specified (base-class) type. The member  
 25 parameter specifies the base-class member that is overridden.

Use this overload to return an **System.Xml.Serialization.XmlAttributes** object that contains objects that override an **System.Xml.Serialization.XmlArrayAttribute** , **System.Xml.Serialization.XmlArrayItemAttribute** , **System.Xml.Serialization.XmlAttributeAttribute** , **System.Xml.Serialization.XmlElementAttribute** , or **System.Xml.Serialization.XmlEnumAttribute** . If the **System.Xml.Serialization.XmlAttributes** object contains for an **System.Xml.Serialization.XmlRootAttribute** or **System.Xml.Serialization.XmlTypeAttribute** , you must use the overload that specifies only the overridden type. The base-class class **System.Type** that is associated with the collection of attributes you want. The name of the overridden member that specifies the **System.Xml.Serialization.XmlAttributes** to return.

Add

[C#] public void Add(Type type, XmlAttributes attributes);

[C++] public: void Add(Type\* type, XmlAttributes\* attributes);

[VB] Public Sub Add(ByVal type As Type, ByVal attributes As XmlAttributes)

[JScript] public function Add(type : Type, attributes : XmlAttributes); Adds an

**System.Xml.Serialization.XmlAttributes** object to the collection of

**System.Xml.Serialization.XmlAttributes** objects.

### *Description*

Adds an **System.Xml.Serialization.XmlAttributes** object to the collection of **System.Xml.Serialization.XmlAttributes** objects. The *type* parameter

1 specifies an object to be overridden by the  
2 **System.Xml.Serialization.XmlAttributes** object.

3 The **System.Xml.Serialization.XmlAttributes** object contains a union of  
4 attribute objects that cause the **System.Xml.Serialization.XmlSerializer** to  
5 override its default serialization behavior for a set of objects. You choose the  
6 attribute objects to place in the **System.Xml.Serialization.XmlAttributes** object,  
7 depending on the particular behaviors you want to override. For example, the  
8 **XmlSerializer** serializes a class member as an XML element by default. If you  
9 want the member to be serialized as an **XmlAttribute** instead, you would create an  
10 **System.Xml.Serialization.XmlAttributeAttribute** , assign it to the  
11 **System.Xml.Serialization.XmlAttributes.XmlAttribute** property of an  
12 **System.Xml.Serialization.XmlAttributes** , and add the  
13 **System.Xml.Serialization.XmlAttributes** object to the  
14 **System.Xml.Serialization.XmlAttributeOverrides** object. The **System.Type** of  
15 the object that will be overridden. An **System.Xml.Serialization.XmlAttributes**  
16 object that represents the overriding attributes.

17 Add

18  
19 [C#] public void Add(Type type, string member, XmlAttributes attributes);  
20 [C++] public: void Add(Type\* type, String\* member, XmlAttributes\* attributes);  
21 [VB] Public Sub Add(ByVal type As Type, ByVal member As String, ByVal  
22 attributes As XmlAttributes)  
23 [JScript] public function Add(type : Type, member : String, attributes :  
24 XmlAttributes);  
25



## Description

Adds an **System.Xml.Serialization.XmlAttributes** object to the collection of **System.Xml.Serialization.XmlAttributes** objects. The *type* parameter specifies an object to be overridden. The *member* parameter specifies the name of a member that will be overridden.

The **System.Xml.Serialization.XmlAttributes** object contains a union of attribute objects that cause the **System.Xml.Serialization.XmlSerializer** to override its default serialization behavior for a set of objects. You choose the attribute objects to place in the **System.Xml.Serialization.XmlAttributes** object, depending on the particular behaviors you want to override. For example, the **XmlSerializer** serializes a class member as an XML element by default. If you want the member to be serialized as an **XmlAttribute** instead, you would create an **System.Xml.Serialization.XmlAttributeAttribute**, assign it to the **System.Xml.Serialization.XmlAttributes.XmlAttribute** property of an **System.Xml.Serialization.XmlAttributes**, and add the **System.Xml.Serialization.XmlAttributes** object to the **System.Xml.Serialization.XmlAttributeOverrides** object. The **System.Type** of the object to override. The name of the member to override. An **System.Xml.Serialization.XmlAttributes** object that represents the overriding attributes.

XmlAttribute class (System.Xml.Serialization)

ToString

### Description

Represents a collection of attribute objects that control how the **System.Xml.Serialization.XmlSerializer** serializes and deserializes an object.

Creating the **System.Xml.Serialization.XmlAttributes** is part of a process that overrides the default way the **System.Xml.Serialization.XmlSerializer** serializes class instances. For example, suppose you want to serialize an object that is created from a DLL which has an inaccessible source. By using the **System.Xml.Serialization.XmlAttributeOverrides**, you can augment or otherwise control how the object is serialized.

**XmlAttributes**

*Example Syntax:*

**ToString**

```
[C#] public XmlAttributes();
```

```
[C++] public: XmlAttributes();
```

```
[VB] Public Sub New()
```

```
[JScript] public function XmlAttributes();
```

Initializes a new instance of the **System.Xml.Serialization.XmlAttributes** class.

### Description

Initializes a new instance of the **System.Xml.Serialization.XmlAttributes** class.

**XmlAttributes**

*Example Syntax:*

ToString

```
[C#] public XmlAttributes(ICustomAttributeProvider provider);  
[C++] public: XmlAttributes(ICustomAttributeProvider* provider);  
[VB] Public Sub New(ByVal provider As ICustomAttributeProvider)  
[JScript] public function XmlAttributes(provider : ICustomAttributeProvider);
```

*Description*

XmlAnyAttribute

ToString

```
[C#] public XmlAnyAttributeAttribute XmlAnyAttribute {get; set;}  
[C++] public: __property XmlAnyAttributeAttribute*  
get_XmlAnyAttribute();public: __property void  
set_XmlAnyAttribute(XmlAnyAttributeAttribute*);  
[VB] Public Property XmlAnyAttribute As XmlAnyAttributeAttribute  
[JScript] public function get XmlAnyAttribute() :  
XmlAnyAttributeAttribute;public function set  
XmlAnyAttribute(XmlAnyAttributeAttribute);
```

*Description*

Gets or sets the **System.Xml.Serialization.XmlAnyAttributeAttribute** to  
override.

XmlAnyElements

ToString

[C#] public XmlAnyElementAttributes XmlAnyElements {get;}

[C++] public: \_\_property XmlAnyElementAttributes\* get\_XmlAnyElements();

[VB] Public ReadOnly Property XmlAnyElements As XmlAnyElementAttributes

[JScript] public function get XmlAnyElements() : XmlAnyElementAttributes;

### *Description*

Gets the collection of

**System.Xml.Serialization.XmlAnyElementAttribute** objects to override.

XmlArray

ToString

[C#] public XmlArrayAttribute XmlArray {get; set;}

[C++] public: \_\_property XmlArrayAttribute\* get\_XmlArray();public: \_\_property

void set\_XmlArray(XmlArrayAttribute\*);

[VB] Public Property XmlArray As XmlArrayAttribute

[JScript] public function get XmlArray() : XmlArrayAttribute;public function set

XmlArray(XmlArrayAttribute);

### *Description*

Gets or sets an object that specifies how the

**System.Xml.Serialization.XmlSerializer** serializes a public field or read/write

property that returns an array.

There are two ways in which a public field or public read/write property that returns an array is serialized by the **System.Xml.Serialization.XmlSerializer** : (1) the default serialization, and (2) the controlled serialization.

**XmlArrayItems**

**ToString**

[C#] public XmlArrayItemAttributes XmlArrayItems {get;}

[C++] public: \_\_property XmlArrayItemAttributes\* get\_XmlArrayItems();

[VB] Public ReadOnly Property XmlArrayItems As XmlArrayItemAttributes

[JScript] public function get XmlArrayItems() : XmlArrayItemAttributes;

### *Description*

Gets or sets a collection of objects that specify how the **System.Xml.Serialization.XmlSerializer** serializes items inserted into an array returned by a public field or read/write property.

The **System.Xml.Serialization.XmlAttributes.XmlArrayItems** property allows you to specify the derived types that can be inserted into an array returned by a public field or public read/write property. For each new type you want the field or property to accept, create an **System.Xml.Serialization.XmlArrayItemAttribute** object and **System.Xml.Serialization.XmlArrayItemAttributes.Add(System.Xml.Serialization.XmlArrayItemAttribute)** it to the **System.Xml.Serialization.XmlArrayItemAttributes** ) returned by the **System.Xml.Serialization.XmlAttributes.XmlArrayItems** property. (The new type must be derived from the type already accepted by the field or property.)

**System.Xml.Serialization.XmlAttributeOverrides.Add(System.Type, System.Xml.Serialization.XmlAttributes)** the **System.Xml.Serialization.XmlAttributes** object to an **System.Xml.Serialization.XmlAttributeOverrides** object, specifying the type of the object containing the field or property, and the name of the field or property. Construct an **System.Xml.Serialization.XmlSerializer** with the **System.Xml.Serialization.XmlAttributeOverrides** object before calling **System.Xml.Serialization.XmlSerializer.Serialize(System.IO.TextWriter, System.Object)** or **System.Xml.Serialization.XmlSerializer.Deserialize(System.IO.Stream)** method.

**XmlAttribute**

**ToString**

[C#] public XmlAttributeAttribute XmlAttribute {get; set;}

[C++] public: \_\_property XmlAttributeAttribute\* get\_XmlAttribute(); public: \_\_property void set\_XmlAttribute(XmlAttributeAttribute\*);

[VB] Public Property XmlAttribute As XmlAttributeAttribute

[JScript] public function get XmlAttribute() : XmlAttributeAttribute; public function set XmlAttribute(XmlAttributeAttribute);

## Description

Gets or sets an object that specifies how the **System.Xml.Serialization.XmlSerializer** serializes a public field or public read/write property as an XML attribute.

1 By default, if no attribute is applied to a public field or public read/write  
2 property, it will be serialized as an XML element. You can also instruct the the  
3 **System.Xml.Serialization.XmlSerializer** to generate an XML attribute by  
4 applying an **System.Xml.Serialization.XmlAttributeAttribute** to the field or  
5 property.

6 XmlChoiceIdentifier

7 ToString

8  
9 [C#] public XmlChoiceIdentifierAttribute XmlChoiceIdentifier {get;}

10 [C++] public: \_\_property XmlChoiceIdentifierAttribute\*  
11 get\_XmlChoiceIdentifier();

12 [VB] Public ReadOnly Property XmlChoiceIdentifier As

13 XmlChoiceIdentifierAttribute

14 [JScript] public function get XmlChoiceIdentifier() :

15 XmlChoiceIdentifierAttribute;

16 XmlDefaultValue

17 ToString

18  
19 [C#] public object XmlDefaultValue {get; set;}

20 [C++] public: \_\_property Object\* get\_XmlDefaultValue();public: \_\_property void  
21 set\_XmlDefaultValue(Object\*);

22 [VB] Public Property XmlDefaultValue As Object

23 [JScript] public function get XmlDefaultValue() : Object;public function set

24 XmlDefaultValue(Object);

## Description

Gets or sets the default value of an XML element or attribute.

You can specify the default value of an XML element or XML attribute by applying a **System.ComponentModel.DefaultValueAttribute** to a member. To examine the result of applying the value, compile the application into a DLL or executable, and pass the resulting file as an argument to the XML Schema Definition tool (XSD.exe). In the XML schema document, a default value is assigned to the **default** attribute. In the example below, the default for the Animal element is "Dogs." To override the default value, create an **System.Object** and assign it to the **System.Xml.Serialization.XmlAttributes.XmlDefaultValue**.

XmlElements

ToString

```
[C#] public XmlElementAttributes XmlElements {get;}
[C++] public: __property XmlElementAttributes* get_XmlElements();
[VB] Public ReadOnly Property XmlElements As XmlElementAttributes
[JScript] public function get XmlElements() : XmlElementAttributes;
```

## Description

Gets or sets a collection of objects that specify how the **System.Xml.Serialization.XmlSerializer** serializes a public field or read/write property as an XML element.

For each overridden member that is serialized as an XML element, you must add a new **System.Xml.Serialization.XmlElementAttribute** to an



**System.Xml.Serialization.XmlElementAttributes** by calling the **System.Xml.Serialization.XmlElementAttributes.Add(System.Xml.Serialization.XmlElementAttribute)** method. By default, an **System.Xml.Serialization.XmlElementAttributes** object is created and assigned to the **System.Xml.Serialization.XmlAttributes.XmlElements** property.

**XmlEnum**

**ToString**

[C#] public XmlEnumAttribute XmlEnum {get; set;}

[C++] public: \_\_property XmlEnumAttribute\* get\_XmlEnum();public: \_\_property void set\_XmlEnum(XmlEnumAttribute\*);

[VB] Public Property XmlEnum As XmlEnumAttribute

[JScript] public function get XmlEnum() : XmlEnumAttribute;public function set XmlEnum(XmlEnumAttribute);

### *Description*

Gets or sets an object that specifies how the **System.Xml.Serialization.XmlSerializer** serializes an enumeration member.

For each identifier you want to override, you must create an **System.Xml.Serialization.XmlAttributes** object, and set the **System.Xml.Serialization.XmlAttributes.XmlEnum** property to an **System.Xml.Serialization.XmlEnumAttribute** that overrides the identifier. Add the **System.Xml.Serialization.XmlAttributes** object to the **System.Xml.Serialization.XmlAttributeOverrides** object, specifying both the

**System.Type** of the class that contains the enumeration, and the overridden member name.

XmlIgnore

ToString

[C#] public bool XmlIgnore {get; set;}

[C++] public: \_\_property bool get\_XmlIgnore();public: \_\_property void set\_XmlIgnore(bool);

[VB] Public Property XmlIgnore As Boolean

[JScript] public function get XmlIgnore() : Boolean;public function set XmlIgnore(Boolean);

### *Description*

Gets or sets a value that specifies whether or not the **System.Xml.Serialization.XmlSerializer** serializes a public field or public read/write property.

By default, all public fields and public read/write properties are serialized by the **System.Xml.Serialization.XmlSerializer** . That is, the value of each public field or property is persisted as an XML element or XML attribute in an XML-document instance.

XmlRoot

ToString

[C#] public XmlRootAttribute XmlRoot {get; set;}

[C++] public: \_\_property XmlRootAttribute\* get\_XmlRoot();public: \_\_property

1 void set\_XmlRoot(XmlRootAttribute\*);

2 [VB] Public Property XmlRoot As XmlRootAttribute

3 [JScript] public function get XmlRoot() : XmlRootAttribute;public function set

4 XmlRoot(XmlRootAttribute);

#### 6 *Description*

7 Gets or sets an object that specifies how the  
8 **System.Xml.Serialization.XmlSerializer** serializes a class as an XML root  
9 element.

10 XmlText

11 ToString

13 [C#] public XmlTextAttribute XmlText {get; set;}

14 [C++] public: \_\_property XmlTextAttribute\* get\_XmlText();public: \_\_property

15 void set\_XmlText(XmlTextAttribute\*);

16 [VB] Public Property XmlText As XmlTextAttribute

17 [JScript] public function get XmlText() : XmlTextAttribute;public function set

18 XmlText(XmlTextAttribute);

#### 20 *Description*

21 Gets or sets an object that instructs the  
22 **System.Xml.Serialization.XmlSerializer** to serialize a public field or public  
23 read/write property as XML text.

24 By default, a public field or public read/write property will be serialized as  
25 an XML element by the **System.Xml.Serialization.XmlSerializer** . However, can

force the field or property to be serialized as XML text by applying an **System.Xml.Serialization.XmlTextAttribute** to the field or property.

XmlType

ToString

[C#] public XmlTypeAttribute XmlType {get; set;}

[C++] public: \_\_property XmlTypeAttribute\* get\_XmlType();public: \_\_property  
void set\_XmlType(XmlTypeAttribute\*);

[VB] Public Property XmlType As XmlTypeAttribute

[JScript] public function get XmlType() : XmlTypeAttribute;public function set  
XmlType(XmlTypeAttribute);

#### *Description*

Gets or sets an object that specifies how the **System.Xml.Serialization.XmlSerializer** serializes a class to which the **System.Xml.Serialization.XmlTypeAttribute** has been applied.

XmlChoiceIdentifierAttribute class (System.Xml.Serialization)

ToString

#### *Description*

Specifies that the member can be further disambiguated by using an enumeration.

The XML schema element definition named **xsi:choice** is used to define a complex element that can contain only one child in an instance (maxoccurs = 1).

That child can be one of several types, and it can have one of several names. Each name is associated with a specific type; however, several names can be associated with the same type. Because of this, an instance of such an element is ambiguous.

For example, consider the following schema fragment that defines such an ambiguous element named **MyChoices** : The

**System.Xml.Serialization.XmlChoiceIdentifierAttribute** allows you to assign a special enumeration value to each instance of the member. You must either create the enumeration yourself, or it can be generated by the tool. The C# code below shows how the **System.Xml.Serialization.XmlChoiceIdentifierAttribute** is applied to an Item field; the

**System.Xml.Serialization.XmlChoiceIdentifierAttribute.MemberName** property identifies the field that contains the enumeration that is further used to disambiguate the choice: public class MyChoice{ [XmlChoice("itemType") public string Item; // Do not serialize this next field: [XmlIgnore] public ItemChoiceType ItemType; } // Do not include this enumeration in the XML schema.

[XmlType(IncludeInSchema = false)] public enum ItemChoiceType{ None, ChoiceOne, ChoiceTwo, } When this code is in place, you can serialize and deserialize this class by setting the itemType field to an appropriate enumeration.

For example to serialize the class, the C# code resembles the following: MyChoice

mc = new MyChoice(); mc.Item = "Item Choice One"; mc.ItemType =

ItemChoiceType.ChoiceOne; When deserializing, the C# code might resemble the

following: MyChoice mc = (MyChoice) myXmlSerializer.Deserialize(myReader);

if(mc.ItemType == ItemChoiceType.ChoiceOne) { // Handle choice one. }

if(mc.ItemType == ItemChoiceType.ChoiceTwo) { // Handle choice two. }

if(mc.ItemType != null) { throw CreateUnknownTypeException(mc.Item); }

There is a second scenario when the **System.Xml.Serialization.XmlChoiceIdentifierAttribute** is used. In the schema below, the member is a field that returns an array of items (maxOccurs="unbounded"). The array can contain objects of the first choice ("D-a-t-a"), and of the second choice ("MoreData"): The resulting class, then, uses a field to return an array of items. And for each item in the array, a corresponding **ItemChoiceType** enumeration must also be found. The matching enumerations are contained in the array returned by **ItemsElementName** field: public class

```
MyChoice { [System.Xml.Serialization.XmlElementAttribute("D-a-t-a",
typeof(string), IsNullable=false)]
[System.Xml.Serialization.XmlElementAttribute("MoreData", typeof(string),
IsNullable=false)]
[System.Xml.Serialization.XmlChoiceIdentifierAttribute("ItemsElementName")]
public string[] Items;
[System.Xml.Serialization.XmlElementAttribute(IsNullable=false)]
[System.Xml.Serialization.XmlIgnoreAttribute()] public ItemsChoiceType[]
ItemsElementName; }
[System.Xml.Serialization.XmlTypeAttribute(IncludeInSchema=false)] public
enum ItemsChoiceType { [System.Xml.Serialization.XmlEnumAttribute("D-a-t-
a")] Data, MoreData, }
```

**XmlChoiceIdentifierAttribute**

*Example Syntax:*

**ToString**

```
[C#] public XmlChoiceIdentifierAttribute();
```

[C++] public: XmlChoiceIdentifierAttribute();

[VB] Public Sub New()

[JScript] public function XmlChoiceIdentifierAttribute(); Initializes a new instance of the **System.Xml.Serialization.XmlChoiceIdentifierAttribute** class.

#### *Description*

Initializes a new instance of the **System.Xml.Serialization.XmlChoiceIdentifierAttribute** class.

XmlChoiceIdentifierAttribute

*Example Syntax:*

ToString

[C#] public XmlChoiceIdentifierAttribute(string name);

[C++] public: XmlChoiceIdentifierAttribute(String\* name);

[VB] Public Sub New(ByVal name As String)

[JScript] public function XmlChoiceIdentifierAttribute(name : String);

#### *Description*

Initializes a new instance of the **System.Xml.Serialization.XmlChoiceIdentifierAttribute** class, and uses the specified value as the name of the member that returns the disambiguating enumeration. The name of the member that will return the enumeration.

MemberName

ToString

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25

```
[C#] public string MemberName {get; set;}  
[C++] public: __property String* get_MemberName();public: __property void  
set_MemberName(String*);  
[VB] Public Property MemberName As String  
[JScript] public function get MemberName() : String;public function set  
MemberName(String);
```

*Description*

Gets or sets the name of the field that returns the enumeration to use when  
disambiguating types.

TypeId  
XmlCodeExporter class (System.Xml.Serialization)  
ToString  
XmlCodeExporter

*Example Syntax:*

ToString

*Description*

XmlCodeExporter  
*Example Syntax:*  
ToString



```

1
2  [C#] public XmlCodeExporter(CodeNamespace codeNamespace,
3  CodeCompileUnit codeCompileUnit);
4  [C++] public: XmlCodeExporter(CodeNamespace* codeNamespace,
5  CodeCompileUnit* codeCompileUnit);
6  [VB] Public Sub New(ByVal codeNamespace As CodeNamespace, ByVal
7  codeCompileUnit As CodeCompileUnit)
8  [JScript] public function XmlCodeExporter(codeNamespace : CodeNamespace,
9  codeCompileUnit : CodeCompileUnit);
10      IncludeMetadata
11      ToString
12
13  [C#] public CodeAttributeDeclarationCollection IncludeMetadata {get;}
14  [C++] public: __property CodeAttributeDeclarationCollection*
15  get_IncludeMetadata();
16  [VB] Public ReadOnly Property IncludeMetadata As
17  CodeAttributeDeclarationCollection
18  [JScript] public function get IncludeMetadata() :
19  CodeAttributeDeclarationCollection;
20
21  Description
22
23      AddMappingMetadata
24
25  [C#] public void AddMappingMetadata(CodeAttributeDeclarationCollection
    
```

```

1 metadata, XmlMemberMapping member, string ns);
2 [C++] public: void AddMappingMetadata(CodeAttributeDeclarationCollection*
3 metadata, XmlMemberMapping* member, String* ns);
4 [VB] Public Sub AddMappingMetadata(ByVal metadata As
5 CodeAttributeDeclarationCollection, ByVal member As XmlMemberMapping,
6 ByVal ns As String)
7 [JScript] public function AddMappingMetadata(metadata :
8 CodeAttributeDeclarationCollection, member : XmlMemberMapping, ns : String);
9

```

### *Description*

#### AddMappingMetadata

```

14 [C#] public void AddMappingMetadata(CodeAttributeDeclarationCollection
15 metadata, XmlTypeMapping mapping, string ns);
16 [C++] public: void AddMappingMetadata(CodeAttributeDeclarationCollection*
17 metadata, XmlTypeMapping* mapping, String* ns);
18 [VB] Public Sub AddMappingMetadata(ByVal metadata As
19 CodeAttributeDeclarationCollection, ByVal mapping As XmlTypeMapping,
20 ByVal ns As String)
21 [JScript] public function AddMappingMetadata(metadata :
22 CodeAttributeDeclarationCollection, mapping : XmlTypeMapping, ns : String);
23

```

### *Description*

## AddMappingMetadata

```

[C#] public void AddMappingMetadata(CodeAttributeDeclarationCollection
metadata, XmlMemberMapping member, string ns, bool forceUseMemberName);

[C++] public: void AddMappingMetadata(CodeAttributeDeclarationCollection*
metadata, XmlMemberMapping* member, String* ns, bool
forceUseMemberName);

[VB] Public Sub AddMappingMetadata(ByVal metadata As
CodeAttributeDeclarationCollection, ByVal member As XmlMemberMapping,
ByVal ns As String, ByVal forceUseMemberName As Boolean)

[JScript] public function AddMappingMetadata(metadata :
CodeAttributeDeclarationCollection, member : XmlMemberMapping, ns : String,
forceUseMemberName : Boolean);
    
```

## *Description*

## ExportMembersMapping

```

[C#] public void ExportMembersMapping(XmlMembersMapping
xmlMembersMapping);

[C++] public: void ExportMembersMapping(XmlMembersMapping*
xmlMembersMapping);

[VB] Public Sub ExportMembersMapping(ByVal xmlMembersMapping As
XmlMembersMapping)

[JScript] public function ExportMembersMapping(xmlMembersMapping :
    
```

XmlMembersMapping);

*Description*

ExportTypeMapping

[C#] public void ExportTypeMapping(XmlTypeMapping xmlTypeMapping);

[C++] public: void ExportTypeMapping(XmlTypeMapping\* xmlTypeMapping);

[VB] Public Sub ExportTypeMapping(ByVal xmlTypeMapping As  
XmlTypeMapping)

[JScript] public function ExportTypeMapping(xmlTypeMapping :  
XmlTypeMapping);

*Description*

XmlElementAttribute class (System.Xml.Serialization)

ToString

*Description*

Indicates that a public field or property represents an XML element--when the **System.Xml.Serialization.XmlSerializer** serializes or deserializes the containing object.

The **System.Xml.Serialization.XmlElementAttribute** belongs to a family of attributes that controls how the **System.Xml.Serialization.XmlSerializer**

serializes, or deserializes, an object. For a complete list of similar attributes, see the **System.Xml.Serialization.XmlSerializer** class.

**XmlElementAttribute**

*Example Syntax:*

**ToString**

[C#] public XmlElementAttribute();

[C++] public: XmlElementAttribute();

[VB] Public Sub New()

[JScript] public function XmlElementAttribute(); Initializes a new instance of the **System.Xml.Serialization.XmlElementAttribute** class.

#### *Description*

Initializes a new instance of the **System.Xml.Serialization.XmlElementAttribute** class.

**XmlElementAttribute**

*Example Syntax:*

**ToString**

[C#] public XmlElementAttribute(string elementName);

[C++] public: XmlElementAttribute(String\* elementName);

[VB] Public Sub New(ByVal elementName As String)

[JScript] public function XmlElementAttribute(elementName : String);

#### *Description*

1        Initializes a new instance of the  
2        **System.Xml.Serialization.XmlElementAttribute** class and specifies the name of  
3        the XML element.

4        By default, the **System.Xml.Serialization.XmlSerializer** uses the member  
5        name as the XML element name when serializing a class instance. For example, a  
6        field named **Vehicle** generates an XML element named **Vehicle** . However, if you  
7        need a different element, such as **Cars** , pass it in the *elementName* parameter. The  
8        XML-element name of the serialized member.

9        XmlElementAttribute

10       *Example Syntax:*

11       ToString

12  
13       [C#] public XmlElementAttribute(Type type);

14       [C++] public: XmlElementAttribute(Type\* type);

15       [VB] Public Sub New(ByVal type As Type)

16       [JScript] public function XmlElementAttribute(type : Type);

#### 17 18       *Description*

19        Initializes a new instance of the  
20        **System.Xml.Serialization.XmlElementAttribute** class, and specifies a derived  
21        type for the member to which the  
22        **System.Xml.Serialization.XmlElementAttribute** is applied--used by the  
23        **System.Xml.Serialization.XmlSerializer** when serializing or deserializing a  
24        containing object.  
25

Use the *type* parameter to specify a type that is derived from a base class. For example, suppose a property named **MyAnimal** returns an **Animal** object. You want to enhance the object, so you create a new class named **Mammal** that inherits from the **Animal** class. To instruct the **System.Xml.Serialization.XmlSerializer** to accept the **Mammal** class when it serializes the **MyAnimal** property, pass the **System.Type** of the **Mammal** class to the constructor. The **System.Type** of an object derived from the member's type.

**XmlElementAttribute**

*Example Syntax:*

**ToString**

```
[C#] public XmlElementAttribute(string elementName, Type type);
[C++] public: XmlElementAttribute(String* elementName, Type* type);
[VB] Public Sub New(ByVal elementName As String, ByVal type As Type)
[JavaScript] public function XmlElementAttribute(elementName : String, type :
Type);
```

### *Description*

Initializes a new instance of the **System.Xml.Serialization.XmlElementAttribute** and specifies the name of the XML element and a derived type for the member to which the **System.Xml.Serialization.XmlElementAttribute** is applied--used when the **System.Xml.Serialization.XmlSerializer** serializes a containing object.

By default, the **System.Xml.Serialization.XmlSerializer** uses the member name as the XML element name when serializing a class instance. For example, a

field named **Vehicle** generates an XML element named **Vehicle** . However, if you need a different element, such as **Cars** , pass it in the *elementName* parameter. The XML-element name of the serialized member. The **System.Type** of an object derived from the member's type.

DataType

ToString

[C#] public string DataType {get; set;}

[C++] public: \_\_property String\* get\_DataType();public: \_\_property void set\_DataType(String\*);

[VB] Public Property DataType As String

[JScript] public function get DataType() : String;public function set DataType(String);

### Description

Gets or sets the XML data type of the generated XML element.

The following table lists the XML data types with their .NET equivalents.

For the XML **date** and **time** data types, the .NET data type consists of a

**System.DateTime** to which the

**System.Xml.Serialization.XmlElementAttribute** has been applied with the

**System.Xml.Serialization.XmlElementAttribute.DataType** set to "date" and

"time", respectively.

ElementName

ToString



1  
2 [C#] public string ElementName {get; set;}

3 [C++] public: \_\_property String\* get\_ElementName();public: \_\_property void  
4 set\_ElementName(String\*);

5 [VB] Public Property ElementName As String

6 [JScript] public function get ElementName() : String;public function set  
7 ElementName(String);

8  
9 *Description*

10 Gets or sets the name of the generated XML element.

11 Specify an

12 **System.Xml.Serialization.XmlArrayItemAttribute.ElementName** if you want  
13 the name of the generated XML element to differ from the member's identifier.

14 Form

15 ToString

16  
17 [C#] public XmlSchemaForm Form {get; set;}

18 [C++] public: \_\_property XmlSchemaForm get\_Form();public: \_\_property void  
19 set\_Form(XmlSchemaForm);

20 [VB] Public Property Form As XmlSchemaForm

21 [JScript] public function get Form() : XmlSchemaForm;public function set  
22 Form(XmlSchemaForm);

23  
24 *Description*

25 Gets or sets a value indicating whether the element is qualified.

The **System.Xml.Serialization.XmlAttributeAttribute.Form** property determines whether an XML element is qualified or unqualified. The **System.Xml.Serialization.XmlAttributeAttribute.Form** property conforms to the 1999 <http://www.w3.org> specification "Namespaces in XML".

IsNullable

ToString

[C#] public bool IsNullable {get; set;}

[C++] public: \_\_property bool get\_IsNullable();public: \_\_property void set\_IsNullable(bool);

[VB] Public Property IsNullable As Boolean

[JScript] public function get IsNullable() : Boolean;public function set IsNullable(Boolean);

### Description

Gets or sets a value indicating whether the **System.Xml.Serialization.XmlSerializer** should serialize a member that is set to **null** into the **xsi:nil** attribute set to **true**.

The XML schema specification for structures allows an XML document to explicitly signal that an element's content is missing. Such an element contains the attribute **xsi:nil** set to **true**. For more information, see the

<http://www.w3.org/TR/xmlschema-1/> specification named *XML Schema Part 1:*

### Structures .

Namespace

ToString

[C#] public string Namespace {get; set;}

[C++] public: \_\_property String\* get\_Namespace();public: \_\_property void

set\_Namespace(String\*);

[VB] Public Property Namespace As String

[JScript] public function get Namespace() : String;public function set

Namespace(String);

### Description

Gets or sets the namespace assigned to the XML element that results when the class is serialized.

The **System.Xml.Serialization.XmlArrayItemAttribute.Namespace** property conforms to the <http://www.w3.org> specification named *Namespaces in XML*.

Type

ToString

[C#] public Type Type {get; set;}

[C++] public: \_\_property Type\* get\_Type();public: \_\_property void

set\_Type(Type\*);

[VB] Public Property Type As Type

[JScript] public function get Type() : Type;public function set Type(Type);

### Description

Gets or sets the object type used to represent the XML element.

1        Use the **System.Xml.Serialization.XmlElementAttribute.Type** property  
2 specify a derived type for a field or property.

3        **TypeId**

4        **XmlElementAttributes** class (**System.Xml.Serialization**)

5        **ToString**

6  
7  
8        *Description*

9        Represents a collection of  
10       **System.Xml.Serialization.XmlElementAttribute** , which the  
11       **System.Xml.Serialization.XmlSerializer** uses to override the default way it  
12       serializes a class.

13       The **System.Xml.Serialization.XmlElementAttributes** is returned by the  
14       **System.Xml.Serialization.XmlAttributes.XmlElements** property of the  
15       **System.Xml.Serialization.XmlAttributes** class. By using the  
16       **System.Xml.Serialization.XmlAttributeOverrides** and the  
17       **System.Xml.Serialization.XmlAttributes** class, you can override the default way  
18       that the **System.Xml.Serialization.XmlSerializer** serializes a class.

19       **XmlElementAttributes**

20       *Example Syntax:*

21       **ToString**

22  
23       [C#] **public XmlElementAttributes();**

24       [C++] **public: XmlElementAttributes();**

1 [VB] Public Sub New()

2 [JScript] public function XmlElementAttributes();

3 Count

4 InnerList

5 Item

6 ToString

9 *Description*

10 Gets or sets an **System.Xml.Serialization.XmlElementAttribute** from the  
11 collection. The zero-based index of the collection member to get or set.

12 List

13 Add

15 [C#] public int Add(XmlElementAttribute attribute);

16 [C++] public: int Add(XmlElementAttribute\* attribute);

17 [VB] Public Function Add(ByVal attribute As XmlElementAttribute) As Integer

18 [JScript] public function Add(attribute : XmlElementAttribute) : int;

20 *Description*

21 Adds an **System.Xml.Serialization.XmlElementAttribute** to the  
22 collection.

23 *Return Value:* The zero-based index of the newly added item. The

24 **System.Xml.Serialization.XmlElementAttribute** to add.

25 Contains

1 [C#] public bool Contains(XmlElementAttribute attribute);

2 [C++] public: bool Contains(XmlElementAttribute\* attribute);

3 [VB] Public Function Contains(ByVal attribute As XmlElementAttribute) As  
4 Boolean

5 [JScript] public function Contains(attribute : XmlElementAttribute) : Boolean;

6  
7  
8 *Description*

9 Gets a value that specifies whether the collections contains the specified  
10 object.

11 *Return Value:* **true** , if the object exists in the collection; otherwise, **false** . The  
12 XmlElementAttribute in question.

13 *CopyTo*

14  
15 [C#] public void CopyTo(XmlElementAttribute[] array, int index);

16 [C++] public: void CopyTo(XmlElementAttribute\* array[], int index);

17 [VB] Public Sub CopyTo(ByVal array() As XmlElementAttribute, ByVal index  
18 As Integer)

19 [JScript] public function CopyTo(array : XmlElementAttribute[], index : int);

20  
21 *Description*

22 Copies the XmlElementAttributes, or a portion of it to a one-dimensional  
23 array. The **System.Xml.Serialization.XmlElementAttribute** array to copy to.  
24 The zero-based index in *array* at which copying begins.

25 *IndexOf*

```

1
2 [C#] public int IndexOf(XmlElementAttribute attribute);
3 [C++] public: int IndexOf(XmlElementAttribute* attribute);
4 [VB] Public Function IndexOf(ByVal attribute As XmlElementAttribute) As
5 Integer
6 [JScript] public function IndexOf(attribute : XmlElementAttribute) : int;
7

```

#### *Description*

Gets the index of the specified

**System.Xml.Serialization.XmlElementAttribute .**

*Return Value:* The zero-based index of the

**System.Xml.Serialization.XmlElementAttribute .**

#### **Insert**

```

15 [C#] public void Insert(int index, XmlElementAttribute attribute);
16 [C++] public: void Insert(int index, XmlElementAttribute* attribute);
17 [VB] Public Sub Insert(ByVal index As Integer, ByVal attribute As
18 XmlElementAttribute)
19 [JScript] public function Insert(index : int, attribute : XmlElementAttribute);
20

```

#### *Description*

Inserts an **System.Xml.Serialization.XmlElementAttribute** into the collection. The zero-based index where the member will be added. The **System.Xml.Serialization.XmlElementAttribute** to insert.

#### **Remove**

```

1
2 [C#] public void Remove(XmlElementAttribute attribute);
3 [C++] public: void Remove(XmlElementAttribute* attribute);
4 [VB] Public Sub Remove(ByVal attribute As XmlElementAttribute)
5 [JScript] public function Remove(attribute : XmlElementAttribute);
6

```

### 7 *Description*

8 Removes the specified object from the collection. The  
9 XmlElementAttribute to remove from the collection.

10 XmlElementEventArgs class (System.Xml.Serialization)

11 ToString

12 Element

13 ToString

14 LineNumber

15 ToString

16 LinePosition

17 ToString

18 ObjectBeingDeserialized

19 ToString

20 XmlElementEventHandler delegate (System.Xml.Serialization)

21 ToString

22 XmlEnumAttribute class (System.Xml.Serialization)

23 ToString



### *Description*

Controls how the **System.Xml.Serialization.XmlSerializer** serializes an enumeration member.

The **System.Xml.Serialization.XmlEnumAttribute** belongs to a family of attributes that controls how the **System.Xml.Serialization.XmlSerializer** serializes, or deserializes, an object. For a complete list of similar attributes, see the **System.Xml.Serialization.XmlSerializer** class.

**XmlEnumAttribute**

*Example Syntax:*

**ToString**

[C#] public XmlEnumAttribute();

[C++] public: XmlEnumAttribute();

[VB] Public Sub New()

[JScript] public function XmlEnumAttribute(); Initializes a new instance of the **System.Xml.Serialization.XmlEnumAttribute** class.

### *Description*

Initializes a new instance of the **System.Xml.Serialization.XmlEnumAttribute** class.

You can use the **System.Xml.Serialization.XmlEnumAttribute.#ctor** to override an existing enumeration.

**XmlEnumAttribute**

*Example Syntax:*

ToString

```
[C#] public XmlEnumAttribute(string name);  
[C++] public: XmlEnumAttribute(String* name);  
[VB] Public Sub New(ByVal name As String)  
[JScript] public function XmlEnumAttribute(name : String);
```

### *Description*

Initializes a new instance of the **System.Xml.Serialization.XmlEnumAttribute** class, and specifies the XML value that the **System.Xml.Serialization.XmlSerializer** generates or recognizes (when it serializes or deserializes the enumeration, respectively).

In your code you can use the word **XmlEnum** instead of the longer **System.Xml.Serialization.XmlEnumAttribute**. The overriding name of the enumeration member.

Name

ToString

```
[C#] public string Name {get; set;}  
[C++] public: __property String* get_Name();public: __property void  
set_Name(String*);  
[VB] Public Property Name As String  
[JScript] public function get Name() : String;public function set Name(String);
```

## Description

Gets or sets the value generated in an XML-document instance when the **System.Xml.Serialization.XmlSerializer** serializes an enumeration, or the value recognized when it deserializes the enumeration member.

Specify the **System.Xml.Serialization.XmlEnumAttribute.Name** when you want the generated XML data to differ from the enumeration identifier.

TypeId

XmlIgnoreAttribute class (System.Xml.Serialization)

ToString

## Description

Instructs the **System.Xml.Serialization.XmlSerializer.Serialize(System.IO.TextWriter, System.Object)** method of the **System.Xml.Serialization.XmlSerializer** not to serialize the public field or public read/write property value.

The **System.Xml.Serialization.XmlIgnoreAttribute** belongs to a family of attributes that controls how the **System.Xml.Serialization.XmlSerializer** serializes, or deserializes, an object. If you apply the **System.Xml.Serialization.XmlIgnoreAttribute** to any member of a class, the **System.Xml.Serialization.XmlSerializer** ignores the member when serializing or deserializing an instance of the class. For a complete list of similar attributes, see the **System.Xml.Serialization.XmlSerializer** class.

XmlIgnoreAttribute

*Example Syntax:*

ToString

[C#] public XmlIgnoreAttribute();

[C++] public: XmlIgnoreAttribute();

[VB] Public Sub New()

[JScript] public function XmlIgnoreAttribute();

### *Description*

Initializes a new instance of the  
**System.Xml.Serialization.XmlIgnoreAttribute** class.

TypeId

XmlAttribute class (System.Xml.Serialization)

ToString

### *Description*

Allows the **System.Xml.Serialization.XmlSerializer** to recognize a  
derived class when it serializes or deserializes an object.

Use the **System.Xml.Serialization.XmlIncludeAttribute** when you call  
the

**System.Xml.Serialization.XmlSerializer.Serialize(System.IO.TextWriter, System.Object)** or

**System.Xml.Serialization.XmlSerializer.Deserialize(System.IO.Stream)**

method of the **System.Xml.Serialization.XmlSerializer** class.

XmlIncludeAttribute

*Example Syntax:*

ToString

[C#] public XmlIncludeAttribute(Type type);

[C++] public: XmlIncludeAttribute(Type\* type);

[VB] Public Sub New(ByVal type As Type)

[JScript] public function XmlIncludeAttribute(type : Type);

*Description*

Initializes a new instance of the **System.Xml.Serialization.XmlIncludeAttribute** class. The **System.Type** of the object to include.

Type

ToString

[C#] public Type Type {get; set;}

[C++] public: \_\_property Type\* get\_Type();public: \_\_property void set\_Type(Type\*);

[VB] Public Property Type As Type

[JScript] public function get Type() : Type;public function set Type(Type);

*Description*

Gets or sets the type of the object to include.

TypeId

MSDN Library

1 XmlMapping class (System.Xml.Serialization)

2 ToString

3  
4  
5 *Description*

6  
7 XmlMemberMapping class (System.Xml.Serialization)

8 ToString

9  
10  
11 *Description*

12  
13 Any

14 ToString

15  
16 [C#] public bool Any {get;}

17 [C++] public: \_\_property bool get\_Any();

18 [VB] Public ReadOnly Property Any As Boolean

19 [JScript] public function get Any() : Boolean;

20  
21 *Description*

22  
23 ElementName

24 ToString

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25

```
[C#] public string ElementName {get;}
[C++] public: __property String* get_ElementName();
[VB] Public ReadOnly Property ElementName As String
[JScript] public function get ElementName() : String;
```

*Description*

MemberName  
ToString

```
[C#] public string MemberName {get;}
[C++] public: __property String* get_MemberName();
[VB] Public ReadOnly Property MemberName As String
[JScript] public function get MemberName() : String;
```

*Description*

Namespace  
ToString

```
[C#] public string Namespace {get;}
[C++] public: __property String* get_Namespace();
[VB] Public ReadOnly Property Namespace As String
[JScript] public function get Namespace() : String;
```

1  
2 *Description*

3  
4       TypeFullName

5       ToString

6  
7 [C#] public string TypeFullName {get;}

8 [C++] public: \_\_property String\* get\_TypeFullName();

9 [VB] Public ReadOnly Property TypeFullName As String

10 [JScript] public function get TypeFullName() : String;

11  
12 *Description*

13  
14       TypeName

15       ToString

16  
17 [C#] public string TypeName {get;}

18 [C++] public: \_\_property String\* get\_TypeName();

19 [VB] Public ReadOnly Property TypeName As String

20 [JScript] public function get TypeName() : String;

21  
22 *Description*

23  
24       TypeNamespace

25       ToString



```
[C#] public string TypeNamespace {get;}
[C++] public: __property String* get_TypeNamespace();
[VB] Public ReadOnly Property TypeNamespace As String
[JScript] public function get TypeNamespace() : String;
```

*Description*

XmlMembersMapping class (System.Xml.Serialization)  
ToString

*Description*

Count  
ToString

```
[C#] public int Count {get;}
[C++] public: __property int get_Count();
[VB] Public ReadOnly Property Count As Integer
[JScript] public function get Count() : int;
```

*Description*

ElementName

ToString

[C#] public string ElementName {get;}

[C++] public: \_\_property String\* get\_ElementName();

[VB] Public ReadOnly Property ElementName As String

[JScript] public function get ElementName() : String;

*Description*

Item

ToString

[C#] public XmlMemberMapping this[int index] {get;}

[C++] public: \_\_property XmlMemberMapping\* get\_Item(int index);

[VB] Public Default ReadOnly Property Item(ByVal index As Integer) As

XmlMemberMapping

[JScript] returnValue = XmlMembersMappingObject.Item(index);

*Description*

Namespace

ToString

[C#] public string Namespace {get;}

[C++] public: \_\_property String\* get\_Namespace();

1 [VB] Public ReadOnly Property Namespace As String

2 [JScript] public function get Namespace() : String;

4 *Description*

6       TypeName

7       ToString

9 [C#] public string TypeName {get;}

10 [C++] public: \_\_property String\* get\_TypeName();

11 [VB] Public ReadOnly Property TypeName As String

12 [JScript] public function get TypeName() : String;

13       TypeNamespace

14       ToString

16 [C#] public string TypeNamespace {get;}

17 [C++] public: \_\_property String\* get\_TypeNamespace();

18 [VB] Public ReadOnly Property TypeNamespace As String

19 [JScript] public function get TypeNamespace() : String;

20       XmlNodeEventArgs class (System.Xml.Serialization)

21       ToString

24 *Description*

Provides data for the  
**System.Xml.Serialization.XmlSerializer.UnknownNode** event.

For more information about handling events, see .

LineNumber

ToString

[C#] public int LineNumber {get;}

[C++] public: \_\_property int get\_LineNumber();

[VB] Public ReadOnly Property LineNumber As Integer

[JScript] public function get LineNumber() : int;

### *Description*

Gets the line number of the unknown XML node.

LinePosition

ToString

[C#] public int LinePosition {get;}

[C++] public: \_\_property int get\_LinePosition();

[VB] Public ReadOnly Property LinePosition As Integer

[JScript] public function get LinePosition() : int;

### *Description*

Gets the position in the line of the unknown XML node.

LocalName

ToString

```

1
2 [C#] public string LocalName {get;}
3 [C++] public: __property String* get_LocalName();
4 [VB] Public ReadOnly Property LocalName As String
5 [JScript] public function get LocalName() : String;
6

```

### Description

Gets the XML local name of the XML node being deserialized.

The **System.Xml.Serialization.XmlNodeEventArgs.LocalName** property returns the local name (also known as a local part) of an XML qualified name. The **System.Xml.Serialization.XmlNodeEventArgs.LocalName** property conforms to the 1999 <http://www.w3.org> specification *Namespaces in XML* .

Name

ToString

```

16 [C#] public string Name {get;}
17 [C++] public: __property String* get_Name();
18 [VB] Public ReadOnly Property Name As String
19 [JScript] public function get Name() : String;
20

```

### Description

Gets the name of the XML node being deserialized.

NamespaceURI

ToString

1  
2 [C#] public string NamespaceURI {get;}

3 [C++] public: \_\_property String\* get \_NamespaceURI();

4 [VB] Public ReadOnly Property NamespaceURI As String

5 [JScript] public function get NamespaceURI() : String;

6  
7 *Description*

8 Gets the namespace URI that is associated with the XML node being  
9 deserialized.

10 NodeType

11 ToString

12  
13 [C#] public XmlNodeType NodeType {get;}

14 [C++] public: \_\_property XmlNodeType get \_NodeType();

15 [VB] Public ReadOnly Property NodeType As XmlNodeType

16 [JScript] public function get NodeType() : XmlNodeType;

17  
18 *Description*

19 Gets the type of the XML node being deserialized.

20 The **System.Xml.XmlNodeType** enumeration returns a description of the  
21 node being deserialized. For example, it returns "Comment" if the node is a  
22 comment.

23 ObjectBeingDeserialized

24 ToString

1  
2 [C#] public object ObjectBeingDeserialized {get;}

3 [C++] public: \_\_property Object\* get\_ObjectBeingDeserialized();

4 [VB] Public ReadOnly Property ObjectBeingDeserialized As Object

5 [JScript] public function get ObjectBeingDeserialized() : Object;

6  
7 *Description*

8 Gets the object being deserialized.

9 Text

10 ToString

11  
12 [C#] public string Text {get;}

13 [C++] public: \_\_property String\* get\_Text();

14 [VB] Public ReadOnly Property Text As String

15 [JScript] public function get Text() : String;

16  
17 *Description*

18 Gets the text of the XML node being deserialized.

19 XmlNodeEventHandler delegate (System.Xml.Serialization)

20 ToString

21  
22  
23 *Description*

24 Represents the method that will handle the

25 **System.Xml.Serialization.XmlSerializer.UnknownNode** event of an

1 **System.Xml.Serialization.XmlSerializer** . The source of the event. An

2 **System.Xml.Serialization.XmlNodeEventArgs** that contains the event data.

3       When you create an **System.Xml.Serialization.XmlNodeEventHandler**  
4 delegate, you identify the method that will handle the event. To associate the event  
5 with your event handler, add an instance of the delegate to the event. The event  
6 handler is called whenever the event occurs, unless you remove the delegate. For  
7 more information about event handler delegates, see .

8       **XmlReflectionImporter** class (System.Xml.Serialization)

9       **ToString**

10  
11  
12 *Description*

13  
14       **XmlReflectionImporter**

15       *Example Syntax:*

16       **ToString**

17  
18 [C#] public XmlReflectionImporter();

19 [C++] public: XmlReflectionImporter();

20 [VB] Public Sub New()

21 [JScript] public function XmlReflectionImporter();

22  
23 *Description*

24  
25       **XmlReflectionImporter**



*Example Syntax:*

ToString

[C#] public XmlReflectionImporter(string defaultNamespace);

[C++] public: XmlReflectionImporter(String\* defaultNamespace);

[VB] Public Sub New(ByVal defaultNamespace As String)

[JScript] public function XmlReflectionImporter(defaultNamespace : String);

*Description*

XmlReflectionImporter

*Example Syntax:*

ToString

[C#] public XmlReflectionImporter(XmlAttributeOverrides attributeOverrides);

[C++] public: XmlReflectionImporter(XmlAttributeOverrides\*  
attributeOverrides);

[VB] Public Sub New(ByVal attributeOverrides As XmlAttributeOverrides)

[JScript] public function XmlReflectionImporter(attributeOverrides :  
XmlAttributeOverrides);

*Description*

XmlReflectionImporter

*Example Syntax:*

## ToString

```
[C#] public XmlReflectionImporter(XmlAttributeOverrides attributeOverrides,  
string defaultNamespace);  
[C++] public: XmlReflectionImporter(XmlAttributeOverrides* attributeOverrides,  
String* defaultNamespace);  
[VB] Public Sub New(ByVal attributeOverrides As XmlAttributeOverrides,  
ByVal defaultNamespace As String)  
[JScript] public function XmlReflectionImporter(attributeOverrides :  
XmlAttributeOverrides, defaultNamespace : String);
```

## *Description*

## ImportMembersMapping

```
[C#] public XmlMembersMapping ImportMembersMapping(string elementName,  
string ns, XmlReflectionMember[] members, bool hasWrapperElement);  
[C++] public: XmlMembersMapping* ImportMembersMapping(String*  
elementName, String* ns, XmlReflectionMember* members[], bool  
hasWrapperElement);  
[VB] Public Function ImportMembersMapping(ByVal elementName As String,  
ByVal ns As String, ByVal members() As XmlReflectionMember, ByVal  
hasWrapperElement As Boolean) As XmlMembersMapping  
[JScript] public function ImportMembersMapping(elementName : String, ns :  
String, members : XmlReflectionMember[], hasWrapperElement : Boolean) :
```

1 XmlMembersMapping;

2  
3 *Description*

4  
5 ImportTypeMapping

6  
7 [C#] public XmlTypeMapping ImportTypeMapping(Type type);

8 [C++] public: XmlTypeMapping\* ImportTypeMapping(Type\* type);

9 [VB] Public Function ImportTypeMapping(ByVal type As Type) As

10 XmlTypeMapping

11 [JScript] public function ImportTypeMapping(type : Type) : XmlTypeMapping;

12  
13 *Description*

14  
15 ImportTypeMapping

16  
17 [C#] public XmlTypeMapping ImportTypeMapping(Type type, string  
18 defaultNamespace);

19 [C++] public: XmlTypeMapping\* ImportTypeMapping(Type\* type, String\*  
20 defaultNamespace);

21 [VB] Public Function ImportTypeMapping(ByVal type As Type, ByVal  
22 defaultNamespace As String) As XmlTypeMapping

23 [JScript] public function ImportTypeMapping(type : Type, defaultNamespace :  
24 String) : XmlTypeMapping;

## Description

### ImportTypeMapping

```
[C#] public XmlTypeMapping ImportTypeMapping(Type type, XmlRootAttribute
root);
[C++] public: XmlTypeMapping* ImportTypeMapping(Type* type,
XmlRootAttribute* root);
[VB] Public Function ImportTypeMapping(ByVal type As Type, ByVal root As
XmlRootAttribute) As XmlTypeMapping
[JScript] public function ImportTypeMapping(type : Type, root :
XmlRootAttribute) : XmlTypeMapping;
```

## Description

### ImportTypeMapping

```
[C#] public XmlTypeMapping ImportTypeMapping(Type type, XmlRootAttribute
root, string defaultNamespace);
[C++] public: XmlTypeMapping* ImportTypeMapping(Type* type,
XmlRootAttribute* root, String* defaultNamespace);
[VB] Public Function ImportTypeMapping(ByVal type As Type, ByVal root As
XmlRootAttribute, ByVal defaultNamespace As String) As XmlTypeMapping
[JScript] public function ImportTypeMapping(type : Type, root :
```

1 XmlRootAttribute, defaultNamespace : String) : XmlTypeMapping;

2  
3 *Description*

4  
5 IncludeType

6  
7 [C#] public void IncludeType(Type type);

8 [C++] public: void IncludeType(Type\* type);

9 [VB] Public Sub IncludeType(ByVal type As Type)

10 [JScript] public function IncludeType(type : Type);

11  
12 *Description*

13  
14 IncludeTypes

15  
16 [C#] public void IncludeTypes(ICustomAttributeProvider provider);

17 [C++] public: void IncludeTypes(ICustomAttributeProvider\* provider);

18 [VB] Public Sub IncludeTypes(ByVal provider As ICustomAttributeProvider)

19 [JScript] public function IncludeTypes(provider : ICustomAttributeProvider);

20  
21 *Description*

22  
23 XmlReflectionMember class (System.Xml.Serialization)

24 ToString

1  
2  
3 *Description*

4  
5 XmlReflectionMember

6 *Example Syntax:*

7 ToString

8  
9 [C#] public XmlReflectionMember();

10 [C++] public: XmlReflectionMember();

11 [VB] Public Sub New()

12 [JScript] public function XmlReflectionMember();

13 IsReturnValue

14 ToString

15  
16 [C#] public bool IsReturnValue {get; set;}

17 [C++] public: \_\_property bool get\_IsReturnValue();public: \_\_property void  
18 set\_IsReturnValue(bool);

19 [VB] Public Property IsReturnValue As Boolean

20 [JScript] public function get IsReturnValue() : Boolean;public function set  
21 IsReturnValue(Boolean);

22  
23 *Description*

24  
25 MemberName

ToString

[C#] public string MemberName {get; set;}

[C++] public: \_\_property String\* get\_MemberName();public: \_\_property void

set\_MemberName(String\*);

[VB] Public Property MemberName As String

[JScript] public function get MemberName() : String;public function set

MemberName(String);

*Description*

MemberType

ToString

[C#] public Type MemberType {get; set;}

[C++] public: \_\_property Type\* get\_MemberType();public: \_\_property void

set\_MemberType(Type\*);

[VB] Public Property MemberType As Type

[JScript] public function get MemberType() : Type;public function set

MemberType(Type);

*Description*

OverrideIsNullable

ToString

[C#] public bool OverrideIsNullable {get; set;}

[C++] public: \_\_property bool get\_OverrideIsNullable();public: \_\_property void  
set\_OverrideIsNullable(bool);

[VB] Public Property OverrideIsNullable As Boolean

[JScript] public function get OverrideIsNullable() : Boolean;public function set  
OverrideIsNullable(Boolean);

SoapAttributes

ToString

[C#] public SoapAttributes SoapAttributes {get; set;}

[C++] public: \_\_property SoapAttributes\* get\_SoapAttributes();public: \_\_property  
void set\_SoapAttributes(SoapAttributes\*);

[VB] Public Property SoapAttributes As SoapAttributes

[JScript] public function get SoapAttributes() : SoapAttributes;public function set  
SoapAttributes(SoapAttributes);

## Description

XmlAttributes

ToString

[C#] public XmlAttributes XmlAttributes {get; set;}

[C++] public: \_\_property XmlAttributes\* get\_XmlAttributes();public: \_\_property  
void set\_XmlAttributes(XmlAttributes\*);



1 [VB] Public Property XmlAttributes As XmlAttributes

2 [JScript] public function get XmlAttributes() : XmlAttributes; public function set

3 XmlAttributes(XmlAttributes);

4  
5 *Description*

6  
7 XmlRootAttribute class (System.Xml.Serialization)

8 ToString

9  
10  
11 *Description*

12 Identifies a class, structure, enumeration, or interface as the root (or top-  
13 level) element of an XML-document instance.

14 The **System.Xml.Serialization.XmlRootAttribute** belongs to a family of  
15 attributes that controls how the **System.Xml.Serialization.XmlSerializer**  
16 serializes, or deserializes, an object. For a complete list of similar attributes, see  
17 the **System.Xml.Serialization.XmlSerializer** class.

18 XmlRootAttribute

19 *Example Syntax:*

20 ToString

21  
22 [C#] public XmlRootAttribute();

23 [C++] public: XmlRootAttribute();

24 [VB] Public Sub New()

25 [JScript] public function XmlRootAttribute(); Initializes a new instance of the

**System.Xml.Serialization.XmlRootAttribute** class, and uses the class name as the name of the XML root element.

#### *Description*

Initializes a new instance of the **System.Xml.Serialization.XmlRootAttribute** class.

**XmlRootAttribute**

*Example Syntax:*

**ToString**

[C#] public XmlRootAttribute(string elementName);

[C++] public: XmlRootAttribute(String\* elementName);

[VB] Public Sub New(ByVal elementName As String)

[JScript] public function XmlRootAttribute(elementName : String);

#### *Description*

Initializes a new instance of the **System.Xml.Serialization.XmlRootAttribute** class, and specifies the name of the XML root element. The name of the XML root element.

**DataType**

**ToString**

[C#] public string DataType {get; set;}

[C++] public: \_\_property String\* get\_DataType();public: \_\_property void

set\_DataType(String\*);

[VB] Public Property DataType As String

[JScript] public function get DataType() : String;public function set

DataType(String);

### Description

Gets or sets the XML data type of the XML root element.

The following table lists the XML data types with their .NET equivalents.

For the XML **date** and **time** data types, the .NET data type consists of a

**System.DateTime** to which the

**System.Xml.Serialization.XmlElementAttribute** has been applied with the

**System.Xml.Serialization.XmlElementAttribute.DataType** set to "date" and

"time", respectively.

ElementName

ToString

[C#] public string ElementName {get; set;}

[C++] public: \_\_property String\* get\_ElementName();public: \_\_property void

set\_ElementName(String\*);

[VB] Public Property ElementName As String

[JScript] public function get ElementName() : String;public function set

ElementName(String);

### Description

Gets or sets the name of the XML element that is generated and recognized

by the **System.Xml.Serialization.XmlSerializer** class's

1 **System.Xml.Serialization.XmlSerializer.Serialize(System.IO.TextWriter, Syst**  
2 **em.Object)** and  
3 **System.Xml.Serialization.XmlSerializer.Deserialize(System.IO.Stream)**  
4 methods, respectively.

5 Specify an  
6 **System.Xml.Serialization.XmlArrayItemAttribute.ElementName** if you want  
7 the name of the generated XML element to differ from the member's identifier.

8 Form

9 ToString

10  
11 [C#] public XmlSchemaForm Form {get; set;}

12 [C++] public: \_\_property XmlSchemaForm get\_Form();public: \_\_property void  
13 set\_Form(XmlSchemaForm);

14 [VB] Public Property Form As XmlSchemaForm

15 [JScript] public function get Form() : XmlSchemaForm;public function set  
16 Form(XmlSchemaForm);

17  
18 *Description*

19 Gets or sets a value indicating whether the name of the XML root element  
20 is qualified.

21 The **System.Xml.Serialization.XmlAttributeAttribute.Form** property  
22 determines whether an XML element name is qualified, based on the  
23 <http://www.w3.org> specification *Namespaces in XML* .

24 IsNullable

25 ToString

```

1
2 [C#] public bool IsNullable {get; set;}
3 [C++] public: __property bool get_IsNullable();public: __property void
4 set_IsNullable(bool);
5 [VB] Public Property IsNullable As Boolean
6 [JScript] public function get IsNullable() : Boolean;public function set
7 IsNullable(Boolean);
8

```

### *Description*

Gets or sets a value indicating whether the **System.Xml.Serialization.XmlSerializer** should serialize a member that is set to **null** into the **xsi:nil** attribute set to **true** .

The XML schema specification for structures allows an XML document to explicitly signal that an element's content is missing. Such an element contains the attribute **xsi:nil** set to **true** . For more information, see the <http://www.w3.org/TR/xmlschema-1/> specification named *XML Schema Part 1: Structures* .

Namespace

ToString

```

21 [C#] public string Namespace {get; set;}
22 [C++] public: __property String* get_Namespace();public: __property void
23 set_Namespace(String*);
24 [VB] Public Property Namespace As String
25 [JScript] public function get Namespace() : String;public function set

```

1 Namespace(String);

3 *Description*

4 Gets or sets the namespace for the XML root element.

5 The **System.Xml.Serialization.XmlArrayItemAttribute.Namespace**  
6 property conforms to the <http://www.w3.org> specification *Namespaces in XML* .

7 TypeId

8 XmlSchemaExporter class (System.Xml.Serialization)

9 ToString

12 *Description*

14 XmlSchemaExporter

15 *Example Syntax:*

16 ToString

18 [C#] public XmlSchemaExporter(XmlSchemas schemas);

19 [C++] public: XmlSchemaExporter(XmlSchemas\* schemas);

20 [VB] Public Sub New(ByVal schemas As XmlSchemas)

21 [JScript] public function XmlSchemaExporter(schemas : XmlSchemas);

23 *Description*

25 ExportAnyType

[C#] public string ExportAnyType(string ns);

[C++] public: String\* ExportAnyType(String\* ns);

[VB] Public Function ExportAnyType(ByVal ns As String) As String

[JScript] public function ExportAnyType(ns : String) : String;

ExportMembersMapping

[C#] public void ExportMembersMapping(XmlMembersMapping

xmlMembersMapping);

[C++] public: void ExportMembersMapping(XmlMembersMapping\*

xmlMembersMapping);

[VB] Public Sub ExportMembersMapping(ByVal xmlMembersMapping As

XmlMembersMapping)

[JScript] public function ExportMembersMapping(xmlMembersMapping :

XmlMembersMapping);

*Description*

ExportTypeMapping

[C#] public XmlQualifiedName ExportTypeMapping(XmlMembersMapping

xmlMembersMapping);

[C++] public: XmlQualifiedName\* ExportTypeMapping(XmlMembersMapping\*

xmlMembersMapping);

[VB] Public Function ExportTypeMapping(ByVal xmlMembersMapping As

```

1 XmlMembersMapping) As XmlQualifiedName
2 [JScript] public function ExportTypeMapping(xmlMembersMapping :
3 XmlMembersMapping) : XmlQualifiedName;
4     ExportTypeMapping
5
6 [C#] public void ExportTypeMapping(XmlTypeMapping xmlTypeMapping);
7 [C++] public: void ExportTypeMapping(XmlTypeMapping* xmlTypeMapping);
8 [VB] Public Sub ExportTypeMapping(ByVal xmlTypeMapping As
9 XmlTypeMapping)
10 [JScript] public function ExportTypeMapping(xmlTypeMapping :
11 XmlTypeMapping);
12

```

### *Description*

XmlSchemaImporter class (System.Xml.Serialization)  
ToString

### *Description*

XmlSchemaImporter

#### *Example Syntax:*

ToString

```

25 [C#] public XmlSchemaImporter(XmlSchemas schemas);

```



1 [C++] public: XmlSchemaImporter(XmlSchemas\* schemas);

2 [VB] Public Sub New(ByVal schemas As XmlSchemas)

3 [JScript] public function XmlSchemaImporter(schemas : XmlSchemas);

4  
5 *Description*

6  
7 XmlSchemaImporter

8 *Example Syntax:*

9 ToString

10  
11 [C#] public XmlSchemaImporter(XmlSchemas schemas, CodeIdentifiers  
12 typeIdentifiers);

13 [C++] public: XmlSchemaImporter(XmlSchemas\* schemas, CodeIdentifiers\*  
14 typeIdentifiers);

15 [VB] Public Sub New(ByVal schemas As XmlSchemas, ByVal typeIdentifiers As  
16 CodeIdentifiers)

17 [JScript] public function XmlSchemaImporter(schemas : XmlSchemas,  
18 typeIdentifiers : CodeIdentifiers);

19  
20 *Description*

21  
22 ImportAnyType

23  
24 [C#] public XmlMembersMapping ImportAnyType(XmlQualifiedName  
25 typeName, string elementName);

```

1 [C++] public: XmlMembersMapping* ImportAnyType(XmlQualifiedName*
2 typeName, String* elementName);
3 [VB] Public Function ImportAnyType(ByVal typeName As XmlQualifiedName,
4 ByVal elementName As String) As XmlMembersMapping
5 [JScript] public function ImportAnyType(typeName : XmlQualifiedName,
6 elementName : String) : XmlMembersMapping;
7     ImportDerivedTypeMapping
8
9 [C#] public XmlTypeMapping ImportDerivedTypeMapping(XmlQualifiedName
10 name, Type baseType);
11 [C++] public: XmlTypeMapping*
12 ImportDerivedTypeMapping(XmlQualifiedName* name, Type* baseType);
13 [VB] Public Function ImportDerivedTypeMapping(ByVal name As
14 XmlQualifiedName, ByVal baseType As Type) As XmlTypeMapping
15 [JScript] public function ImportDerivedTypeMapping(name : XmlQualifiedName,
16 baseType : Type) : XmlTypeMapping;
17
18 Description
19
20     ImportDerivedTypeMapping
21
22 [C#] public XmlTypeMapping ImportDerivedTypeMapping(XmlQualifiedName
23 name, Type baseType, bool baseTypeCanBeIndirect);
24 [C++] public: XmlTypeMapping*
25 ImportDerivedTypeMapping(XmlQualifiedName* name, Type* baseType, bool

```

```

1 baseTypeCanBeIndirect);
2 [VB] Public Function ImportDerivedTypeMapping(ByVal name As
3 XmlQualifiedName, ByVal baseType As Type, ByVal baseTypeCanBeIndirect As
4 Boolean) As XmlTypeMapping
5 [JScript] public function ImportDerivedTypeMapping(name : XmlQualifiedName,
6 baseType : Type, baseTypeCanBeIndirect : Boolean) : XmlTypeMapping;
7     ImportMembersMapping
8
9 [C#] public XmlMembersMapping ImportMembersMapping(XmlQualifiedName
10 name);
11 [C++] public: XmlMembersMapping*
12 ImportMembersMapping(XmlQualifiedName* name);
13 [VB] Public Function ImportMembersMapping(ByVal name As
14 XmlQualifiedName) As XmlMembersMapping
15 [JScript] public function ImportMembersMapping(name : XmlQualifiedName) :
16 XmlMembersMapping;
17
18 Description
19
20     ImportMembersMapping
21
22 [C#] public XmlMembersMapping
23 ImportMembersMapping(XmlQualifiedName[] names);
24 [C++] public: XmlMembersMapping*
25 ImportMembersMapping(XmlQualifiedName* names[]);

```

```

1 [VB] Public Function ImportMembersMapping(ByVal names() As
2 XmlQualifiedName) As XmlMembersMapping
3 [JScript] public function ImportMembersMapping(names : XmlQualifiedName[])
4 : XmlMembersMapping;

```

## *Description*

### ImportMembersMapping

```

10 [C#] public XmlMembersMapping
11 ImportMembersMapping(XmlQualifiedName[] names, Type baseType, bool
12 baseTypeCanBeIndirect);
13 [C++] public: XmlMembersMapping*
14 ImportMembersMapping(XmlQualifiedName* names[], Type* baseType, bool
15 baseTypeCanBeIndirect);
16 [VB] Public Function ImportMembersMapping(ByVal names() As
17 XmlQualifiedName, ByVal baseType As Type, ByVal baseTypeCanBeIndirect As
18 Boolean) As XmlMembersMapping
19 [JScript] public function ImportMembersMapping(names : XmlQualifiedName[],
20 baseType : Type, baseTypeCanBeIndirect : Boolean) : XmlMembersMapping;
21 ImportTypeMapping
22
23 [C#] public XmlTypeMapping ImportTypeMapping(XmlQualifiedName name);
24 [C++] public: XmlTypeMapping* ImportTypeMapping(XmlQualifiedName*
25 name);

```

1 [VB] Public Function ImportTypeMapping(ByVal name As XmlQualifiedName)

2 As XmlTypeMapping

3 [JScript] public function ImportTypeMapping(name : XmlQualifiedName) :

4 XmlTypeMapping;

5  
6 *Description*

7  
8 XmlSchemas class (System.Xml.Serialization)

9 ToString

10  
11  
12 *Description*

13 Represents the collection of XML schemas.

14 XmlSchemas

15 *Example Syntax:*

16 ToString

17  
18 [C#] public XmlSchemas();

19 [C++] public: XmlSchemas();

20 [VB] Public Sub New()

21 [JScript] public function XmlSchemas();

22 Count

23 InnerList

24 Item

25 ToString

*Description*

Item

ToString

[C#] public XmlSchema this[string ns] {get;}

[C++] public: \_\_property XmlSchema\* get\_Item(String\* ns);

[VB] Public Default ReadOnly Property Item(ByVal ns As String) As XmlSchema

[JScript] returnValue = XmlSchemasObject.Item(ns);

*Description*

List

Add

[C#] public int Add(XmlSchema schema);

[C++] public: int Add(XmlSchema\* schema);

[VB] Public Function Add(ByVal schema As XmlSchema) As Integer

[JScript] public function Add(schema : XmlSchema) : int;

*Description*

Add

[C#] public void Add(XmlSchemas schemas);

[C++] public: void Add(XmlSchemas\* schemas);

[VB] Public Sub Add(ByVal schemas As XmlSchemas)

[JScript] public function Add(schemas : XmlSchemas);

### *Description*

#### Contains

[C#] public bool Contains(XmlSchema schema);

[C++] public: bool Contains(XmlSchema\* schema);

[VB] Public Function Contains(ByVal schema As XmlSchema) As Boolean

[JScript] public function Contains(schema : XmlSchema) : Boolean;

### *Description*

#### CopyTo

[C#] public void CopyTo(XmlSchema[] array, int index);

[C++] public: void CopyTo(XmlSchema\* array[], int index);

[VB] Public Sub CopyTo(ByVal array() As XmlSchema, ByVal index As Integer)

[JScript] public function CopyTo(array : XmlSchema[], index : int);

### *Description*

## Find

```
[C#] public object Find(XmlQualifiedName name, Type type);  
[C++] public: Object* Find(XmlQualifiedName* name, Type* type);  
[VB] Public Function Find(ByVal name As XmlQualifiedName, ByVal type As  
Type) As Object  
[JScript] public function Find(name : XmlQualifiedName, type : Type) : Object;
```

### *Description*

## IndexOf

```
[C#] public int IndexOf(XmlSchema schema);  
[C++] public: int IndexOf(XmlSchema* schema);  
[VB] Public Function IndexOf(ByVal schema As XmlSchema) As Integer  
[JScript] public function IndexOf(schema : XmlSchema) : int;
```

### *Description*

## Insert

```
[C#] public void Insert(int index, XmlSchema schema);  
[C++] public: void Insert(int index, XmlSchema* schema);  
[VB] Public Sub Insert(ByVal index As Integer, ByVal schema As XmlSchema)
```



1 [JScript] public function Insert(index : int, schema : XmlSchema);

3 *Description*

5       IsDataSet

7 [C#] public static bool IsDataSet(XmlSchema schema);

8 [C++] public: static bool IsDataSet(XmlSchema\* schema);

9 [VB] Public Shared Function IsDataSet(ByVal schema As XmlSchema) As

10 Boolean

11 [JScript] public static function IsDataSet(schema : XmlSchema) : Boolean;

13 *Description*

15       OnClear

17 [C#] protected override void OnClear();

18 [C++] protected: void OnClear();

19 [VB] Overrides Protected Sub OnClear()

20 [JScript] protected override function OnClear();

22 *Description*

24       OnInsert

1  
2 [C#] protected override void OnInsert(int index, object value);  
3 [C++] protected: void OnInsert(int index, Object\* value);  
4 [VB] Overrides Protected Sub OnInsert(ByVal index As Integer, ByVal value As  
5 Object)  
6 [JScript] protected override function OnInsert(index : int, value : Object);  
7

## 8 *Description*

### 10 OnRemove

11  
12 [C#] protected override void OnRemove(int index, object value);  
13 [C++] protected: void OnRemove(int index, Object\* value);  
14 [VB] Overrides Protected Sub OnRemove(ByVal index As Integer, ByVal value  
15 As Object)  
16 [JScript] protected override function OnRemove(index : int, value : Object);  
17

## 18 *Description*

### 20 OnSet

21  
22 [C#] protected override void OnSet(int index, object oldValue, object newValue);  
23 [C++] protected: void OnSet(int index, Object\* oldValue, Object\* newValue);  
24 [VB] Overrides Protected Sub OnSet(ByVal index As Integer, ByVal oldValue As  
25 Object, ByVal newValue As Object)

[JScript] protected override function OnSet(index : int, oldValue : Object,  
newValue : Object);

*Description*

Remove

[C#] public void Remove(XmlSchema schema);

[C++] public: void Remove(XmlSchema\* schema);

[VB] Public Sub Remove(ByVal schema As XmlSchema)

[JScript] public function Remove(schema : XmlSchema);

*Description*

XmlSerializationCollectionFixupCallback delegate  
(System.Xml.Serialization)

ToString

*Description*

XmlSerializationFixupCallback delegate (System.Xml.Serialization)  
ToString

*Description*

XmlSerializationReadCallback delegate (System.Xml.Serialization)

ToString

*Description*

XmlSerializationReader class (System.Xml.Serialization)

ToString

*Description*

XmlSerializationReader

*Example Syntax:*

ToString

[C#] protected XmlSerializationReader();

[C++] protected: XmlSerializationReader();

[VB] Protected Sub New()

[JScript] protected function XmlSerializationReader();

Document

ToString

[C#] protected XmlDocument Document {get;}

[C++] protected: \_\_property XmlDocument\* get\_Document();

[VB] Protected ReadOnly Property Document As XmlDocument

[JScript] protected function get Document() : XmlDocument;

*Description*

Reader

ToString

[C#] protected XmlReader Reader {get;}

[C++] protected: \_\_property XmlReader\* get\_Reader();

[VB] Protected ReadOnly Property Reader As XmlReader

[JScript] protected function get Reader() : XmlReader;

*Description*

AddFixup

[C#] protected void AddFixup(XmlSerializationReader.CollectionFixup fixup);

[C++] protected: void AddFixup(XmlSerializationReader.CollectionFixup\*  
fixup);

[VB] Protected Sub AddFixup(ByVal fixup As

XmlSerializationReader.CollectionFixup)

[JScript] protected function AddFixup(fixup :

XmlSerializationReader.CollectionFixup);

### *Description*

#### AddFixup

[C#] protected void AddFixup(XmlSerializationReader.Fixup fixup);

[C++] protected: void AddFixup(XmlSerializationReader.Fixup\* fixup);

[VB] Protected Sub AddFixup(ByVal fixup As XmlSerializationReader.Fixup)

[JScript] protected function AddFixup(fixup : XmlSerializationReader.Fixup);

### *Description*

#### AddReadCallback

[C#] protected void AddReadCallback(string name, string ns, Type type,

XmlSerializationReadCallback read);

[C++] protected: void AddReadCallback(String\* name, String\* ns, Type\* type,

XmlSerializationReadCallback\* read);

[VB] Protected Sub AddReadCallback(ByVal name As String, ByVal ns As

String, ByVal type As Type, ByVal read As XmlSerializationReadCallback)

[JScript] protected function AddReadCallback(name : String, ns : String, type :

Type, read : XmlSerializationReadCallback);

*Description*

AddTarget

[C#] protected void AddTarget(string id, object o);

[C++] protected: void AddTarget(String\* id, Object\* o);

[VB] Protected Sub AddTarget(ByVal id As String, ByVal o As Object)

[JScript] protected function AddTarget(id : String, o : Object);

*Description*

CreateAbstractTypeException

[C#] protected Exception CreateAbstractTypeException(string name, string ns);

[C++] protected: Exception\* CreateAbstractTypeException(String\* name, String\* ns);

[VB] Protected Function CreateAbstractTypeException(ByVal name As String, ByVal ns As String) As Exception

[JScript] protected function CreateAbstractTypeException(name : String, ns : String) : Exception;

*Description*

CreateReadOnlyCollectionException

[C#] protected Exception CreateReadOnlyCollectionException(string name);  
 [C++] protected: Exception\* CreateReadOnlyCollectionException(String\* name);  
 [VB] Protected Function CreateReadOnlyCollectionException(ByVal name As String) As Exception  
 [JScript] protected function CreateReadOnlyCollectionException(name : String) : Exception;

### *Description*

#### CreateUnknownConstantException

[C#] protected Exception CreateUnknownConstantException(string value, Type enumType);  
 [C++] protected: Exception\* CreateUnknownConstantException(String\* value, Type\* enumType);  
 [VB] Protected Function CreateUnknownConstantException(ByVal value As String, ByVal enumType As Type) As Exception  
 [JScript] protected function CreateUnknownConstantException(value : String, enumType : Type) : Exception;

### *Description*

#### CreateUnknownNodeException



```
1
2 [C#] protected Exception CreateUnknownNodeException();
3 [C++] protected: Exception* CreateUnknownNodeException();
4 [VB] Protected Function CreateUnknownNodeException() As Exception
5 [JScript] protected function CreateUnknownNodeException() : Exception;
```

## 6 *Description*

### 7 *CreateUnknownTypeException*

```
10
11 [C#] protected Exception CreateUnknownTypeException(XmlQualifiedName
12 type);
13 [C++] protected: Exception* CreateUnknownTypeException(XmlQualifiedName*
14 type);
15 [VB] Protected Function CreateUnknownTypeException(ByVal type As
16 XmlQualifiedName) As Exception
17 [JScript] protected function CreateUnknownTypeException(type :
18 XmlQualifiedName) : Exception;
```

## 19 *Description*

### 20 *EnsureArrayIndex*

```
21
22
23
24 [C#] protected Array EnsureArrayIndex(Array a, int index, Type elementType);
25 [C++] protected: Array* EnsureArrayIndex(Array* a, int index, Type*
```

elementType);

[VB] Protected Function EnsureArrayIndex(ByVal a As Array, ByVal index As Integer, ByVal elementType As Type) As Array

[JScript] protected function EnsureArrayIndex(a : Array, index : int, elementType : Type) : Array;

### *Description*

#### FixupArrayRefs

[C#] protected void FixupArrayRefs(object fixup);

[C++] protected: void FixupArrayRefs(Object\* fixup);

[VB] Protected Sub FixupArrayRefs(ByVal fixup As Object)

[JScript] protected function FixupArrayRefs(fixup : Object);

### *Description*

#### GetArrayLength

[C#] protected int GetArrayLength(string name, string ns);

[C++] protected: int GetArrayLength(String\* name, String\* ns);

[VB] Protected Function GetArrayLength(ByVal name As String, ByVal ns As String) As Integer

[JScript] protected function GetArrayLength(name : String, ns : String) : int;

*Description*

GetNullAttr

[C#] protected bool GetNullAttr();

[C++] protected: bool GetNullAttr();

[VB] Protected Function GetNullAttr() As Boolean

[JScript] protected function GetNullAttr() : Boolean;

*Description*

GetTarget

[C#] protected object GetTarget(string id);

[C++] protected: Object\* GetTarget(String\* id);

[VB] Protected Function GetTarget(ByVal id As String) As Object

[JScript] protected function GetTarget(id : String) : Object;

*Description*

GetXsiType

[C#] protected XmlQualifiedName GetXsiType();

[C++] protected: XmlQualifiedName\* GetXsiType();

[VB] Protected Function GetXsiType() As XmlQualifiedName  
 [JScript] protected function GetXsiType() : XmlQualifiedName;

*Description*

InitCallbacks

[C#] protected abstract void InitCallbacks();  
 [C++] protected: virtual void InitCallbacks() = 0;  
 [VB] MustOverride Protected Sub InitCallbacks()  
 [JScript] protected abstract function InitCallbacks();

*Description*

InitIDs

[C#] protected abstract void InitIDs();  
 [C++] protected: virtual void InitIDs() = 0;  
 [VB] MustOverride Protected Sub InitIDs()  
 [JScript] protected abstract function InitIDs();

*Description*

IsXmlnsAttribute

1  
2 [C#] protected bool IsXmlnsAttribute(string name);

3 [C++] protected: bool IsXmlnsAttribute(String\* name);

4 [VB] Protected Function IsXmlnsAttribute(ByVal name As String) As Boolean

5 [JScript] protected function IsXmlnsAttribute(name : String) : Boolean;

6  
7 *Description*

8  
9 ParseWsdArrayType

10  
11 [C#] protected void ParseWsdArrayType(XmlAttribute attr);

12 [C++] protected: void ParseWsdArrayType(XmlAttribute\* attr);

13 [VB] Protected Sub ParseWsdArrayType(ByVal attr As XmlAttribute)

14 [JScript] protected function ParseWsdArrayType(attr : XmlAttribute);

15  
16 *Description*

17  
18 ReadElementQualifiedName

19  
20 [C#] protected XmlQualifiedName ReadElementQualifiedName();

21 [C++] protected: XmlQualifiedName\* ReadElementQualifiedName();

22 [VB] Protected Function ReadElementQualifiedName() As XmlQualifiedName

23 [JScript] protected function ReadElementQualifiedName() : XmlQualifiedName;

24  
25 *Description*

## ReadEndElement

[C#] protected void ReadEndElement();

[C++] protected: void ReadEndElement();

[VB] Protected Sub ReadEndElement()

[JScript] protected function ReadEndElement();

### *Description*

## ReadNull

[C#] protected bool ReadNull();

[C++] protected: bool ReadNull();

[VB] Protected Function ReadNull() As Boolean

[JScript] protected function ReadNull() : Boolean;

### *Description*

## ReadNullableQualifiedName

[C#] protected XmlQualifiedName ReadNullableQualifiedName();

[C++] protected: XmlQualifiedName\* ReadNullableQualifiedName();

[VB] Protected Function ReadNullableQualifiedName() As XmlQualifiedName

[JScript] protected function ReadNullableQualifiedName() : XmlQualifiedName;

MSI-865US.APP 509\*324\*9256 lee@hayes plc

*Description*

ReadNullableString

[C#] protected string ReadNullableString();  
[C++] protected: String\* ReadNullableString();  
[VB] Protected Function ReadNullableString() As String  
[JScript] protected function ReadNullableString() : String;

*Description*

ReadReference

[C#] protected bool ReadReference(out string fixupReference);  
[C++] protected: bool ReadReference(String\*\* fixupReference);  
[VB] Protected Function ReadReference(ByRef fixupReference As String) As Boolean  
[JScript] protected function ReadReference(fixupReference : String) : Boolean;

*Description*

ReadReferencedElement

[C#] protected object ReadReferencedElement();

1 [C++] protected: Object\* ReadReferencedElement();

2 [VB] Protected Function ReadReferencedElement() As Object

3 [JScript] protected function ReadReferencedElement() : Object;

4  
5 *Description*

6  
7 ReadReferencedElement

8  
9 [C#] protected object ReadReferencedElement(string name, string ns);

10 [C++] protected: Object\* ReadReferencedElement(String\* name, String\* ns);

11 [VB] Protected Function ReadReferencedElement(ByVal name As String, ByVal  
12 ns As String) As Object

13 [JScript] protected function ReadReferencedElement(name : String, ns : String) :  
14 Object;

15  
16 *Description*

17  
18 ReadReferencedElements

19  
20 [C#] protected void ReadReferencedElements();

21 [C++] protected: void ReadReferencedElements();

22 [VB] Protected Sub ReadReferencedElements()

23 [JScript] protected function ReadReferencedElements();

24  
25 *Description*



## ReadReferencingElement

[C#] protected object ReadReferencingElement(out string fixupReference);  
[C++] protected: Object\* ReadReferencingElement(String\*\* fixupReference);  
[VB] Protected Function ReadReferencingElement(ByRef fixupReference As String) As Object  
[JScript] protected function ReadReferencingElement(fixupReference : String) : Object;

### *Description*

## ReadReferencingElement

[C#] protected object ReadReferencingElement(string name, string ns, out string fixupReference);  
[C++] protected: Object\* ReadReferencingElement(String\* name, String\* ns, String\*\* fixupReference);  
[VB] Protected Function ReadReferencingElement(ByVal name As String, ByVal ns As String, ByRef fixupReference As String) As Object  
[JScript] protected function ReadReferencingElement(name : String, ns : String, fixupReference : String) : Object;

### *Description*

## ReadSerializable

[C#] protected IXmlSerializable ReadSerializable(IXmlSerializable serializable);

[C++] protected: IXmlSerializable\* ReadSerializable(IXmlSerializable\* serializable);

[VB] Protected Function ReadSerializable(ByVal serializable As IXmlSerializable) As IXmlSerializable

[JScript] protected function ReadSerializable(serializable : IXmlSerializable) : IXmlSerializable;

### *Description*

## ReadString

[C#] protected string ReadString(string value);

[C++] protected: String\* ReadString(String\* value);

[VB] Protected Function ReadString(ByVal value As String) As String

[JScript] protected function ReadString(value : String) : String;

### *Description*

## ReadTypedPrimitive

[C#] protected object ReadTypedPrimitive(XmlQualifiedName type);

[C++] protected: Object\* ReadTypedPrimitive(XmlQualifiedName\* type);

[VB] Protected Function ReadTypedPrimitive(ByVal type As

XmlQualifiedName) As Object

[JScript] protected function ReadTypedPrimitive(type : XmlQualifiedName) :

Object;

*Description*

ReadXmlNode

[C#] protected XmlNode ReadXmlNode(bool wrapped);

[C++] protected: XmlNode\* ReadXmlNode(bool wrapped);

[VB] Protected Function ReadXmlNode(ByVal wrapped As Boolean) As

XmlNode

[JScript] protected function ReadXmlNode(wrapped : Boolean) : XmlNode;

*Description*

ShrinkArray

[C#] protected Array ShrinkArray(Array a, int length, Type elementType, bool  
isNullable);

[C++] protected: Array\* ShrinkArray(Array\* a, int length, Type\* elementType,  
bool isNullable);

[VB] Protected Function ShrinkArray(ByVal a As Array, ByVal length As

Integer, ByVal elementType As Type, ByVal isNullable As Boolean) As Array

[JScript] protected function ShrinkArray(a : Array, length : int, elementType : Type, isNullable : Boolean) : Array;

### Description

#### ToByteArray

[C#] protected static byte[] ToByteArray(string value);

[C++] protected: static unsigned char ToByteArray(String\* value) \_\_gc[];

[VB] Protected Shared Function ToByteArray(ByVal value As String) As Byte()

[JScript] protected static function ToByteArray(value : String) : Byte[];

### Description

#### ToByteArrayBase64

[C#] protected static byte[] ToByteArrayBase64(string value);

[C++] protected: static unsigned char ToByteArrayBase64(String\* value) \_\_gc[];

[VB] Protected Shared Function ToByteArrayBase64(ByVal value As String) As Byte()

[JScript] protected static function ToByteArrayBase64(value : String) : Byte[];

### Description

#### ToByteArrayHex

```

1
2 [C#] protected static byte[] ToByteArrayHex(string value);
3 [C++] protected: static unsigned char ToByteArrayHex(String* value) __gc[];
4 [VB] Protected Shared Function ToByteArrayHex(ByVal value As String) As
5 Byte()
6 [JScript] protected static function ToByteArrayHex(value : String) : Byte[];
7

```

### *Description*

#### ToChar

```

12 [C#] protected static char ToChar(string value);
13 [C++] protected: static __wchar_t ToChar(String* value);
14 [VB] Protected Shared Function ToChar(ByVal value As String) As Char
15 [JScript] protected static function ToChar(value : String) : Char;
16

```

### *Description*

#### ToDate

```

21 [C#] protected static DateTime ToDate(string value);
22 [C++] protected: static DateTime ToDate(String* value);
23 [VB] Protected Shared Function ToDate(ByVal value As String) As DateTime
24 [JScript] protected static function ToDate(value : String) : DateTime;
25

```

*Description*

**DateTime**

[C#] protected static DateTime ToDateTime(string value);  
 [C++] protected: static DateTime ToDateTime(String\* value);  
 [VB] Protected Shared Function ToDateTime(ByVal value As String) As  
 DateTime  
 [JScript] protected static function ToDateTime(value : String) : DateTime;

*Description*

**ToInt**

[C#] protected static long ToInt(string value, Hashtable h, string typeName);  
 [C++] protected: static \_\_int64 ToInt(String\* value, Hashtable\* h, String\*  
 typeName);  
 [VB] Protected Shared Function ToInt(ByVal value As String, ByVal h As  
 Hashtable, ByVal typeName As String) As Long  
 [JScript] protected static function ToInt(value : String, h : Hashtable, typeName  
 : String) : long;

*Description*

## ToTime

[C#] protected static DateTime ToTime(string value);

[C++] protected: static DateTime ToTime(String\* value);

[VB] Protected Shared Function ToTime(ByVal value As String) As DateTime

[JScript] protected static function ToTime(value : String) : DateTime;

### *Description*

## ToXmlQualifiedName

[C#] protected XmlQualifiedName ToXmlQualifiedName(string value);

[C++] protected: XmlQualifiedName\* ToXmlQualifiedName(String\* value);

[VB] Protected Function ToXmlQualifiedName(ByVal value As String) As

XmlQualifiedName

[JScript] protected function ToXmlQualifiedName(value : String) :

XmlQualifiedName;

### *Description*

## UnknownAttribute

[C#] protected void UnknownAttribute(object o, XmlAttribute attr);

[C++] protected: void UnknownAttribute(Object\* o, XmlAttribute\* attr);

[VB] Protected Sub UnknownAttribute(ByVal o As Object, ByVal attr As

1 XmlAttribute)

2 [JScript] protected function UnknownAttribute(o : Object, attr : XmlAttribute);

4 *Description*

6 UnknownElement

8 [C#] protected void UnknownElement(object o, XmlElement elem);

9 [C++] protected: void UnknownElement(Object\* o, XmlElement\* elem);

10 [VB] Protected Sub UnknownElement(ByVal o As Object, ByVal elem As  
11 XmlElement)

12 [JScript] protected function UnknownElement(o : Object, elem : XmlElement);

13 UnknownNode

15 [C#] protected void UnknownNode(object o);

16 [C++] protected: void UnknownNode(Object\* o);

17 [VB] Protected Sub UnknownNode(ByVal o As Object)

18 [JScript] protected function UnknownNode(o : Object);

20 *Description*

22 UnreferencedObject

24 [C#] protected void UnreferencedObject(string id, object o);

25 [C++] protected: void UnreferencedObject(String\* id, Object\* o);



1 [VB] Protected Sub UnreferencedObject(ByVal id As String, ByVal o As Object)

2 [JScript] protected function UnreferencedObject(id : String, o : Object);

3 XmlSerializationWriteCallback delegate (System.Xml.Serialization)

4 UnreferencedObject

7 *Description*

9 XmlSerializationWriter class (System.Xml.Serialization)

10 UnreferencedObject

13 *Description*

15 XmlSerializationWriter

16 *Example Syntax:*

17 UnreferencedObject

19 [C#] protected XmlSerializationWriter();

20 [C++] protected: XmlSerializationWriter();

21 [VB] Protected Sub New()

22 [JScript] protected function XmlSerializationWriter();

23 Namespaces

24 UnreferencedObject

1  
2 [C#] protected ArrayList Namespaces {get; set;}

3 [C++] protected: \_\_property ArrayList\* get\_Namespaces();protected: \_\_property

4 void set\_Namespaces(ArrayList\*);

5 [VB] Protected Property Namespaces As ArrayList

6 [JScript] protected function get Namespaces() : ArrayList;protected function set

7 Namespaces(ArrayList);

8  
9 *Description*

10  
11 Writer

12 UnreferencedObject

13  
14 [C#] protected XmlWriter Writer {get; set;}

15 [C++] protected: \_\_property XmlWriter\* get\_Writer();protected: \_\_property void

16 set\_Writer(XmlWriter\*);

17 [VB] Protected Property Writer As XmlWriter

18 [JScript] protected function get Writer() : XmlWriter;protected function set

19 Writer(XmlWriter);

20  
21 *Description*

22  
23 AddWriteCallback

24  
25 [C#] protected void AddWriteCallback(Type type, string typeName, string

```

1 typeNs, XmlSerializationWriteCallback callback);
2 [C++] protected: void AddWriteCallback(Type* type, String* typeName, String*
3 typeNs, XmlSerializationWriteCallback* callback);
4 [VB] Protected Sub AddWriteCallback(ByVal type As Type, ByVal typeName As
5 String, ByVal typeNs As String, ByVal callback As
6 XmlSerializationWriteCallback)
7 [JScript] protected function AddWriteCallback(type : Type, typeName : String,
8 typeNs : String, callback : XmlSerializationWriteCallback);
9

```

## *Description*

### CreateMismatchChoiceException

```

14 [C#] protected Exception CreateMismatchChoiceException(string value, string
15 elementName, string enumValue);

```

```

16 [C++] protected: Exception* CreateMismatchChoiceException(String* value,
17 String* elementName, String* enumValue);

```

```

18 [VB] Protected Function CreateMismatchChoiceException(ByVal value As
19 String, ByVal elementName As String, ByVal enumValue As String) As
20 Exception

```

```

21 [JScript] protected function CreateMismatchChoiceException(value : String,
22 elementName : String, enumValue : String) : Exception;

```

### CreateUnknownTypeException

```

25 [C#] protected Exception CreateUnknownTypeException(object o);

```

[C++] protected: Exception\* CreateUnknownTypeException(Object\* o);

[VB] Protected Function CreateUnknownTypeException(ByVal o As Object) As  
Exception

[JScript] protected function CreateUnknownTypeException(o : Object) :  
Exception;

### *Description*

CreateUnknownTypeException

[C#] protected Exception CreateUnknownTypeException(Type type);

[C++] protected: Exception\* CreateUnknownTypeException(Type\* type);

[VB] Protected Function CreateUnknownTypeException(ByVal type As Type) As  
Exception

[JScript] protected function CreateUnknownTypeException(type : Type) :  
Exception;

### *Description*

FromArray

[C#] protected static string FromByteArray(byte[] value);

[C++] protected: static String\* FromByteArray(unsigned char value \_\_gc[]);

[VB] Protected Shared Function FromByteArray(ByVal value() As Byte) As  
String

[JScript] protected static function FromByteArray(value : Byte[]) : String;

### *Description*

#### FromByteArrayBase64

[C#] protected static string FromByteArrayBase64(byte[] value);

[C++] protected: static String\* FromByteArrayBase64(unsigned char value  
\_\_gc[]);

[VB] Protected Shared Function FromByteArrayBase64(ByVal value() As Byte)  
As String

[JScript] protected static function FromByteArrayBase64(value : Byte[]) : String;

#### FromByteArrayHex

[C#] protected static string FromByteArrayHex(byte[] value);

[C++] protected: static String\* FromByteArrayHex(unsigned char value \_\_gc[]);

[VB] Protected Shared Function FromByteArrayHex(ByVal value() As Byte) As  
String

[JScript] protected static function FromByteArrayHex(value : Byte[]) : String;

#### FromChar

[C#] protected static string FromChar(char value);

[C++] protected: static String\* FromChar(\_\_wchar\_t value);

[VB] Protected Shared Function FromChar(ByVal value As Char) As String

[JScript] protected static function FromChar(value : Char) : String;

## FromDate

```
[C#] protected static string FromDate(DateTime value);
[C++] protected: static String* FromDate(DateTime value);
[VB] Protected Shared Function FromDate(ByVal value As DateTime) As String
[JScript] protected static function FromDate(value : DateTime) : String;
```

## FromDateTime

```
[C#] protected static string FromDateTime(DateTime value);
[C++] protected: static String* FromDateTime(DateTime value);
[VB] Protected Shared Function FromDateTime(ByVal value As DateTime) As
String
[JScript] protected static function FromDateTime(value : DateTime) : String;
```

## FromEnum

```
[C#] protected static string FromEnum(long value, string[] values, long[] ids);
[C++] protected: static String* FromEnum(__int64 value, String* values __gc[],
__int64 ids __gc[]);
[VB] Protected Shared Function FromEnum(ByVal value As Long, ByVal
values() As String, ByVal ids() As Long) As String
[JScript] protected static function FromEnum(value : long, values : String[], ids :
long[]) : String;
```

## FromTime

```
[C#] protected static string FromTime(DateTime value);
```

1 [C++] protected: static String\* FromTime(DateTime value);

2 [VB] Protected Shared Function FromTime(ByVal value As DateTime) As String

3 [JScript] protected static function FromTime(value : DateTime) : String;

4     FromXmlQualifiedName

5

6 [C#] protected string FromXmlQualifiedName(XmlQualifiedName

7 xmlQualifiedName);

8 [C++] protected: String\* FromXmlQualifiedName(XmlQualifiedName\*

9 xmlQualifiedName);

10 [VB] Protected Function FromXmlQualifiedName(ByVal xmlQualifiedName As

11 XmlQualifiedName) As String

12 [JScript] protected function FromXmlQualifiedName(xmlQualifiedName :

13 XmlQualifiedName) : String;

14

### *Description*

#### InitCallbacks

19 [C#] protected abstract void InitCallbacks();

20 [C++] protected: virtual void InitCallbacks() = 0;

21 [VB] MustOverride Protected Sub InitCallbacks()

22 [JScript] protected abstract function InitCallbacks();

23

### *Description*

## TopLevelElement

[C#] protected void TopLevelElement();  
 [C++] protected: void TopLevelElement();  
 [VB] Protected Sub TopLevelElement()  
 [JScript] protected function TopLevelElement();

## WriteAttribute

[C#] protected void WriteAttribute(string localName, string value);  
 [C++] protected: void WriteAttribute(String\* localName, String\* value);  
 [VB] Protected Sub WriteAttribute(ByVal localName As String, ByVal value As String)  
 [JScript] protected function WriteAttribute(localName : String, value : String);

## Description

## WriteAttribute

[C#] protected void WriteAttribute(string localName, string ns, string value);  
 [C++] protected: void WriteAttribute(String\* localName, String\* ns, String\* value);  
 [VB] Protected Sub WriteAttribute(ByVal localName As String, ByVal ns As String, ByVal value As String)  
 [JScript] protected function WriteAttribute(localName : String, ns : String, value : String);



*Description*

WriteAttribute

[C#] protected void WriteAttribute(string prefix, string localName, string ns, string value);

[C++] protected: void WriteAttribute(String\* prefix, String\* localName, String\* ns, String\* value);

[VB] Protected Sub WriteAttribute(ByVal prefix As String, ByVal localName As String, ByVal ns As String, ByVal value As String)

[JScript] protected function WriteAttribute(prefix : String, localName : String, ns : String, value : String);

*Description*

WriteElementEncoded

[C#] protected void WriteElementEncoded(XmlNode node, string name, string ns, bool isNullable, bool any);

[C++] protected: void WriteElementEncoded(XmlNode\* node, String\* name, String\* ns, bool isNullable, bool any);

[VB] Protected Sub WriteElementEncoded(ByVal node As XmlNode, ByVal name As String, ByVal ns As String, ByVal isNullable As Boolean, ByVal any As Boolean)

1 [JScript] protected function WriteElementEncoded(node : XmlNode, name :  
2 String, ns : String, isNullable : Boolean, any : Boolean);

3 WriteElementLiteral

4  
5 [C#] protected void WriteElementLiteral(XmlNode node, string name, string ns,  
6 bool isNullable, bool any);

7 [C++] protected: void WriteElementLiteral(XmlNode\* node, String\* name,  
8 String\* ns, bool isNullable, bool any);

9 [VB] Protected Sub WriteElementLiteral(ByVal node As XmlNode, ByVal name  
10 As String, ByVal ns As String, ByVal isNullable As Boolean, ByVal any As  
11 Boolean)

12 [JScript] protected function WriteElementLiteral(node : XmlNode, name : String,  
13 ns : String, isNullable : Boolean, any : Boolean);

14 WriteElementQualifiedName

15  
16 [C#] protected void WriteElementQualifiedName(string localName,  
17 XmlQualifiedName value);

18 [C++] protected: void WriteElementQualifiedName(String\* localName,  
19 XmlQualifiedName\* value);

20 [VB] Protected Sub WriteElementQualifiedName(ByVal localName As String,  
21 ByVal value As XmlQualifiedName)

22 [JScript] protected function WriteElementQualifiedName(localName : String,  
23 value : XmlQualifiedName);

24  
25 *Description*

## WriteElementQualifiedName

```
[C#] protected void WriteElementQualifiedName(string localName, string ns,
XmlQualifiedName value);
[C++] protected: void WriteElementQualifiedName(String* localName, String*
ns, XmlQualifiedName* value);
[VB] Protected Sub WriteElementQualifiedName(ByVal localName As String,
ByVal ns As String, ByVal value As XmlQualifiedName)
[JScript] protected function WriteElementQualifiedName(localName : String, ns :
String, value : XmlQualifiedName);
```

## WriteElementQualifiedName

```
[C#] protected void WriteElementQualifiedName(string localName,
XmlQualifiedName value, XmlQualifiedName xsiType);
[C++] protected: void WriteElementQualifiedName(String* localName,
XmlQualifiedName* value, XmlQualifiedName* xsiType);
[VB] Protected Sub WriteElementQualifiedName(ByVal localName As String,
ByVal value As XmlQualifiedName, ByVal xsiType As XmlQualifiedName)
[JScript] protected function WriteElementQualifiedName(localName : String,
value : XmlQualifiedName, xsiType : XmlQualifiedName);
```

## WriteElementQualifiedName

```
[C#] protected void WriteElementQualifiedName(string localName, string ns,
XmlQualifiedName value, XmlQualifiedName xsiType);
```

1 [C++] protected: void WriteElementQualifiedName(String\* localName, String\*  
2 ns, XmlQualifiedName\* value, XmlQualifiedName\* xsiType);  
3 [VB] Protected Sub WriteElementQualifiedName(ByVal localName As String,  
4 ByVal ns As String, ByVal value As XmlQualifiedName, ByVal xsiType As  
5 XmlQualifiedName)  
6 [JScript] protected function WriteElementQualifiedName(localName : String, ns :  
7 String, value : XmlQualifiedName, xsiType : XmlQualifiedName);

#### WriteElementString

10 [C#] protected void WriteElementString(string localName, string value);  
11 [C++] protected: void WriteElementString(String\* localName, String\* value);  
12 [VB] Protected Sub WriteElementString(ByVal localName As String, ByVal  
13 value As String)  
14 [JScript] protected function WriteElementString(localName : String, value :  
15 String);

#### *Description*

#### WriteElementString

21 [C#] protected void WriteElementString(string localName, string ns, string value);  
22 [C++] protected: void WriteElementString(String\* localName, String\* ns, String\*  
23 value);  
24 [VB] Protected Sub WriteElementString(ByVal localName As String, ByVal ns  
25 As String, ByVal value As String)

1 [JScript] protected function WriteElementString(localName : String, ns : String,  
2 value : String);

3  
4 *Description*

5  
6 WriteElementString

7  
8 [C#] protected void WriteElementString(string localName, string value,  
9 XmlQualifiedName xsiType);

10 [C++] protected: void WriteElementString(String\* localName, String\* value,  
11 XmlQualifiedName\* xsiType);

12 [VB] Protected Sub WriteElementString(ByVal localName As String, ByVal  
13 value As String, ByVal xsiType As XmlQualifiedName)

14 [JScript] protected function WriteElementString(localName : String, value :  
15 String, xsiType : XmlQualifiedName);

16 WriteElementString

17  
18 [C#] protected void WriteElementString(string localName, string ns, string value,  
19 XmlQualifiedName xsiType);

20 [C++] protected: void WriteElementString(String\* localName, String\* ns, String\*  
21 value, XmlQualifiedName\* xsiType);

22 [VB] Protected Sub WriteElementString(ByVal localName As String, ByVal ns  
23 As String, ByVal value As String, ByVal xsiType As XmlQualifiedName)

24 [JScript] protected function WriteElementString(localName : String, ns : String,  
25 value : String, xsiType : XmlQualifiedName);

## WriteElementStringRaw

```
[C#] protected void WriteElementStringRaw(string localName, string value);  
[C++] protected: void WriteElementStringRaw(String* localName, String* value);  
[VB] Protected Sub WriteElementStringRaw(ByVal localName As String, ByVal  
value As String)  
[JScript] protected function WriteElementStringRaw(localName : String, value :  
String);
```

## WriteElementStringRaw

```
[C#] protected void WriteElementStringRaw(string localName, string ns, string  
value);  
[C++] protected: void WriteElementStringRaw(String* localName, String* ns,  
String* value);  
[VB] Protected Sub WriteElementStringRaw(ByVal localName As String, ByVal  
ns As String, ByVal value As String)  
[JScript] protected function WriteElementStringRaw(localName : String, ns :  
String, value : String);
```

## WriteElementStringRaw

```
[C#] protected void WriteElementStringRaw(string localName, string value,  
XmlQualifiedName xsiType);  
[C++] protected: void WriteElementStringRaw(String* localName, String* value,  
XmlQualifiedName* xsiType);  
[VB] Protected Sub WriteElementStringRaw(ByVal localName As String, ByVal
```

1 value As String, ByVal xsiType As XmlQualifiedName)  
2 [JScript] protected function WriteElementStringRaw(localName : String, value :  
3 String, xsiType : XmlQualifiedName);

4 WriteElementStringRaw

5  
6 [C#] protected void WriteElementStringRaw(string localName, string ns, string  
7 value, XmlQualifiedName xsiType);

8 [C++] protected: void WriteElementStringRaw(String\* localName, String\* ns,  
9 String\* value, XmlQualifiedName\* xsiType);

10 [VB] Protected Sub WriteElementStringRaw(ByVal localName As String, ByVal  
11 ns As String, ByVal value As String, ByVal xsiType As XmlQualifiedName)

12 [JScript] protected function WriteElementStringRaw(localName : String, ns :  
13 String, value : String, xsiType : XmlQualifiedName);

14 WriteEmptyTag

15  
16 [C#] protected void WriteEmptyTag(string name, string ns);

17 [C++] protected: void WriteEmptyTag(String\* name, String\* ns);

18 [VB] Protected Sub WriteEmptyTag(ByVal name As String, ByVal ns As String)

19 [JScript] protected function WriteEmptyTag(name : String, ns : String);

20  
21 *Description*

22  
23 WriteEndElement

24  
25 [C#] protected void WriteEndElement();

```

1  [C++] protected: void WriteEndElement();
2  [VB] Protected Sub WriteEndElement()
3  [JScript] protected function WriteEndElement();
4      WriteEndElement
5
6  [C#] protected void WriteEndElement(object o);
7  [C++] protected: void WriteEndElement(Object* o);
8  [VB] Protected Sub WriteEndElement(ByVal o As Object)
9  [JScript] protected function WriteEndElement(o : Object);
10     WriteId
11
12  [C#] protected void WriteId(object o);
13  [C++] protected: void WriteId(Object* o);
14  [VB] Protected Sub WriteId(ByVal o As Object)
15  [JScript] protected function WriteId(o : Object);
16
17  Description
18
19      WriteNullableQualifiedNameEncoded
20
21  [C#] protected void WriteNullableQualifiedNameEncoded(string name, string ns,
22  XmlQualifiedName value, XmlQualifiedName xsiType);
23  [C++] protected: void WriteNullableQualifiedNameEncoded(String* name,
24  String* ns, XmlQualifiedName* value, XmlQualifiedName* xsiType);
25  [VB] Protected Sub WriteNullableQualifiedNameEncoded(ByVal name As String,

```



```

1 ByVal ns As String, ByVal value As XmlQualifiedName, ByVal xsiType As
2 XmlQualifiedName)
3 [JScript] protected function WriteNullableQualifiedNameEncoded(name : String,
4 ns : String, value : XmlQualifiedName, xsiType : XmlQualifiedName);
5     WriteNullableQualifiedNameLiteral
6
7 [C#] protected void WriteNullableQualifiedNameLiteral(string name, string ns,
8 XmlQualifiedName value);
9 [C++] protected: void WriteNullableQualifiedNameLiteral(String* name, String*
10 ns, XmlQualifiedName* value);
11 [VB] Protected Sub WriteNullableQualifiedNameLiteral(ByVal name As String,
12 ByVal ns As String, ByVal value As XmlQualifiedName)
13 [JScript] protected function WriteNullableQualifiedNameLiteral(name : String, ns
14 : String, value : XmlQualifiedName);
15     WriteNullableStringEncoded
16
17 [C#] protected void WriteNullableStringEncoded(string name, string ns, string
18 value, XmlQualifiedName xsiType);
19 [C++] protected: void WriteNullableStringEncoded(String* name, String* ns,
20 String* value, XmlQualifiedName* xsiType);
21 [VB] Protected Sub WriteNullableStringEncoded(ByVal name As String, ByVal
22 ns As String, ByVal value As String, ByVal xsiType As XmlQualifiedName)
23 [JScript] protected function WriteNullableStringEncoded(name : String, ns :
24 String, value : String, xsiType : XmlQualifiedName);
25     WriteNullableStringEncodedRaw

```

```

1
2 [C#] protected void WriteNullableStringEncodedRaw(string name, string ns,
3 string value, XmlQualifiedName xsiType);
4 [C++] protected: void WriteNullableStringEncodedRaw(String* name, String* ns,
5 String* value, XmlQualifiedName* xsiType);
6 [VB] Protected Sub WriteNullableStringEncodedRaw(ByVal name As String,
7 ByVal ns As String, ByVal value As String, ByVal xsiType As
8 XmlQualifiedName)
9 [JScript] protected function WriteNullableStringEncodedRaw(name : String, ns :
10 String, value : String, xsiType : XmlQualifiedName);
11     WriteNullableStringLiteral
12
13 [C#] protected void WriteNullableStringLiteral(string name, string ns, string
14 value);
15 [C++] protected: void WriteNullableStringLiteral(String* name, String* ns,
16 String* value);
17 [VB] Protected Sub WriteNullableStringLiteral(ByVal name As String, ByVal ns
18 As String, ByVal value As String)
19 [JScript] protected function WriteNullableStringLiteral(name : String, ns : String,
20 value : String);
21     WriteNullableStringLiteralRaw
22
23 [C#] protected void WriteNullableStringLiteralRaw(string name, string ns, string
24 value);
25 [C++] protected: void WriteNullableStringLiteralRaw(String* name, String* ns,

```

```

1 String* value);
2 [VB] Protected Sub WriteNullableStringLiteralRaw(ByVal name As String,
3 ByVal ns As String, ByVal value As String)
4 [JScript] protected function WriteNullableStringLiteralRaw(name : String, ns :
5 String, value : String);
6     WriteNullTagEncoded
7
8 [C#] protected void WriteNullTagEncoded(string name);
9 [C++] protected: void WriteNullTagEncoded(String* name);
10 [VB] Protected Sub WriteNullTagEncoded(ByVal name As String)
11 [JScript] protected function WriteNullTagEncoded(name : String);
12     WriteNullTagEncoded
13
14 [C#] protected void WriteNullTagEncoded(string name, string ns);
15 [C++] protected: void WriteNullTagEncoded(String* name, String* ns);
16 [VB] Protected Sub WriteNullTagEncoded(ByVal name As String, ByVal ns As
17 String)
18 [JScript] protected function WriteNullTagEncoded(name : String, ns : String);
19     WriteNullTagLiteral
20
21 [C#] protected void WriteNullTagLiteral(string name);
22 [C++] protected: void WriteNullTagLiteral(String* name);
23 [VB] Protected Sub WriteNullTagLiteral(ByVal name As String)
24 [JScript] protected function WriteNullTagLiteral(name : String);
25     WriteNullTagLiteral

```

1  
2 [C#] protected void WriteNullTagLiteral(string name, string ns);

3 [C++] protected: void WriteNullTagLiteral(String\* name, String\* ns);

4 [VB] Protected Sub WriteNullTagLiteral(ByVal name As String, ByVal ns As  
5 String)

6 [JScript] protected function WriteNullTagLiteral(name : String, ns : String);

7  
8 *Description*

9  
10 WritePotentiallyReferencingElement

11  
12 [C#] protected void WritePotentiallyReferencingElement(string n, string ns, object  
13 o);

14 [C++] protected: void WritePotentiallyReferencingElement(String\* n, String\* ns,  
15 Object\* o);

16 [VB] Protected Sub WritePotentiallyReferencingElement(ByVal n As String,  
17 ByVal ns As String, ByVal o As Object)

18 [JScript] protected function WritePotentiallyReferencingElement(n : String, ns :  
19 String, o : Object);

20  
21 *Description*

22  
23 WritePotentiallyReferencingElement

24  
25 [C#] protected void WritePotentiallyReferencingElement(string n, string ns, object

```

1 o, Type ambientType);
2 [C++] protected: void WritePotentiallyReferencingElement(String* n, String* ns,
3 Object* o, Type* ambientType);
4 [VB] Protected Sub WritePotentiallyReferencingElement(ByVal n As String,
5 ByVal ns As String, ByVal o As Object, ByVal ambientType As Type)
6 [JScript] protected function WritePotentiallyReferencingElement(n : String, ns :
7 String, o : Object, ambientType : Type);
8

```

### *Description*

#### WritePotentiallyReferencingElement

```

13 [C#] protected void WritePotentiallyReferencingElement(string n, string ns, object
14 o, Type ambientType, bool suppressReference);
15 [C++] protected: void WritePotentiallyReferencingElement(String* n, String* ns,
16 Object* o, Type* ambientType, bool suppressReference);
17 [VB] Protected Sub WritePotentiallyReferencingElement(ByVal n As String,
18 ByVal ns As String, ByVal o As Object, ByVal ambientType As Type, ByVal
19 suppressReference As Boolean)
20 [JScript] protected function WritePotentiallyReferencingElement(n : String, ns :
21 String, o : Object, ambientType : Type, suppressReference : Boolean);
22

```

#### WritePotentiallyReferencingElement

```

24 [C#] protected void WritePotentiallyReferencingElement(string n, string ns, object
25 o, Type ambientType, bool suppressReference, bool isNullable);

```

[C++] protected: void WritePotentiallyReferencingElement(String\* n, String\* ns, Object\* o, Type\* ambientType, bool suppressReference, bool isNullable);

[VB] Protected Sub WritePotentiallyReferencingElement(ByVal n As String, ByVal ns As String, ByVal o As Object, ByVal ambientType As Type, ByVal suppressReference As Boolean, ByVal isNullable As Boolean)

[JScript] protected function WritePotentiallyReferencingElement(n : String, ns : String, o : Object, ambientType : Type, suppressReference : Boolean, isNullable : Boolean);

#### WriteReferencedElements

[C#] protected void WriteReferencedElements();

[C++] protected: void WriteReferencedElements();

[VB] Protected Sub WriteReferencedElements()

[JScript] protected function WriteReferencedElements();

#### *Description*

#### WriteReferencingElement

[C#] protected void WriteReferencingElement(string n, string ns, object o);

[C++] protected: void WriteReferencingElement(String\* n, String\* ns, Object\* o);

[VB] Protected Sub WriteReferencingElement(ByVal n As String, ByVal ns As String, ByVal o As Object)

[JScript] protected function WriteReferencingElement(n : String, ns : String, o : Object);

1  
2 *Description*

3  
4 WriteReferencingElement

5  
6 [C#] protected void WriteReferencingElement(string n, string ns, object o, bool  
7 isNullable);

8 [C++] protected: void WriteReferencingElement(String\* n, String\* ns, Object\* o,  
9 bool isNullable);

10 [VB] Protected Sub WriteReferencingElement(ByVal n As String, ByVal ns As  
11 String, ByVal o As Object, ByVal isNullable As Boolean)

12 [JScript] protected function WriteReferencingElement(n : String, ns : String, o :  
13 Object, isNullable : Boolean);

14 WriteSerializable

15  
16 [C#] protected void WriteSerializable(IXmlSerializable serializable, string name,  
17 string ns, bool isNullable);

18 [C++] protected: void WriteSerializable(IXmlSerializable\* serializable, String\*  
19 name, String\* ns, bool isNullable);

20 [VB] Protected Sub WriteSerializable(ByVal serializable As IXmlSerializable,  
21 ByVal name As String, ByVal ns As String, ByVal isNullable As Boolean)

22 [JScript] protected function WriteSerializable(serializable : IXmlSerializable,  
23 name : String, ns : String, isNullable : Boolean);

24  
25 *Description*

## WriteStartDocument

[C#] protected void WriteStartDocument();  
 [C++] protected: void WriteStartDocument();  
 [VB] Protected Sub WriteStartDocument()  
 [JScript] protected function WriteStartDocument();

### *Description*

## WriteStartElement

[C#] protected void WriteStartElement(string name);  
 [C++] protected: void WriteStartElement(String\* name);  
 [VB] Protected Sub WriteStartElement(ByVal name As String)  
 [JScript] protected function WriteStartElement(name : String);

### *Description*

## WriteStartElement

[C#] protected void WriteStartElement(string name, string ns);  
 [C++] protected: void WriteStartElement(String\* name, String\* ns);  
 [VB] Protected Sub WriteStartElement(ByVal name As String, ByVal ns As String)



1 [JScript] protected function WriteStartElement(name : String, ns : String);

3 *Description*

5 WriteStartElement

7 [C#] protected void WriteStartElement(string name, string ns, bool writePrefixed);

8 [C++] protected: void WriteStartElement(String\* name, String\* ns, bool  
9 writePrefixed);

10 [VB] Protected Sub WriteStartElement(ByVal name As String, ByVal ns As  
11 String, ByVal writePrefixed As Boolean)

12 [JScript] protected function WriteStartElement(name : String, ns : String,  
13 writePrefixed : Boolean);

14 WriteStartElement

16 [C#] protected void WriteStartElement(string name, string ns, object o);

17 [C++] protected: void WriteStartElement(String\* name, String\* ns, Object\* o);

18 [VB] Protected Sub WriteStartElement(ByVal name As String, ByVal ns As  
19 String, ByVal o As Object)

20 [JScript] protected function WriteStartElement(name : String, ns : String, o :  
21 Object);

22 WriteStartElement

24 [C#] protected void WriteStartElement(string name, string ns, object o, bool  
25 writePrefixed);

[C++] protected: void WriteStartElement(String\* name, String\* ns, Object\* o,  
bool writePrefixed);

[VB] Protected Sub WriteStartElement(ByVal name As String, ByVal ns As  
String, ByVal o As Object, ByVal writePrefixed As Boolean)

[JScript] protected function WriteStartElement(name : String, ns : String, o :  
Object, writePrefixed : Boolean);

### WriteTypedPrimitive

[C#] protected void WriteTypedPrimitive(string name, string ns, object o, bool  
xsiType);

[C++] protected: void WriteTypedPrimitive(String\* name, String\* ns, Object\* o,  
bool xsiType);

[VB] Protected Sub WriteTypedPrimitive(ByVal name As String, ByVal ns As  
String, ByVal o As Object, ByVal xsiType As Boolean)

[JScript] protected function WriteTypedPrimitive(name : String, ns : String, o :  
Object, xsiType : Boolean);

### Description

### WriteXmlAttribute

[C#] protected void WriteXmlAttribute(XmlNode node);

[C++] protected: void WriteXmlAttribute(XmlNode\* node);

[VB] Protected Sub WriteXmlAttribute(ByVal node As XmlNode)

[JScript] protected function WriteXmlAttribute(node : XmlNode);

## WriteXsiType

[C#] protected void WriteXsiType(string name, string ns);

[C++] protected: void WriteXsiType(String\* name, String\* ns);

[VB] Protected Sub WriteXsiType(ByVal name As String, ByVal ns As String)

[JScript] protected function WriteXsiType(name : String, ns : String);

### *Description*

XmlSerializer class (System.Xml.Serialization)

## WriteXsiType

### *Description*

Serializes and deserializes objects into and from XML-document instances.

The **System.Xml.Serialization.XmlSerializer** enables you to control how objects are encoded into XML. Optionally, you can use the to generate classes that the serializer can encode into XML that adheres to a given XSD schema document.

XML serialization is the process of converting an object's public properties and fields to a serial format (in this case, XML) for storage or transport.

Deserialization recreates the object-in its original state-from the XML output. You can thus think of serialization as a way of saving the state of an object into a stream or buffer. For example, ASP.Net web services uses the

**System.Xml.Serialization.XmlSerializer** to encode web service messages.

## XmlSerializer

*Example Syntax:*

WriteXsiType

[C#] protected XmlSerializer();

[C++] protected: XmlSerializer();

[VB] Protected Sub New()

[JScript] protected function XmlSerializer(); Constructs a new instance of the

**System.Xml.Serialization.XmlSerializer** class.

#### *Description*

Constructs a new instance of the **System.Xml.Serialization.XmlSerializer** class.

XmlSerializer

*Example Syntax:*

WriteXsiType

[C#] public XmlSerializer(Type type);

[C++] public: XmlSerializer(Type\* type);

[VB] Public Sub New(ByVal type As Type)

[JScript] public function XmlSerializer(type : Type);

#### *Description*

Initializes a new instance of the **System.Xml.Serialization.XmlSerializer** class that can serialize objects of the specified type into XML-document instances, and vice versa.

Commonly, an application defines several classes that the **System.Xml.Serialization.XmlSerializer** converts into a single XML-instance document. However, the **System.Xml.Serialization.XmlSerializer** needs only to know only one type--the type of the class that represents the XML root element. The **System.Xml.Serialization.XmlSerializer** automatically serializes all subordinate class instances. Similarly, only the type of the XML root element is needed for deserialization. The type of the object that this **System.Xml.Serialization.XmlSerializer** can serialize.

XmlSerializer

*Example Syntax:*

WriteXsiType

[C#] public XmlSerializer(XmlTypeMapping xmlTypeMapping);

[C++] public: XmlSerializer(XmlTypeMapping\* xmlTypeMapping);

[VB] Public Sub New(ByVal xmlTypeMapping As XmlTypeMapping)

[JScript] public function XmlSerializer(xmlTypeMapping : XmlTypeMapping);

The **System.Xml.Serialization.XmlTypeMapping** .

XmlSerializer

*Example Syntax:*

WriteXsiType

[C#] public XmlSerializer(Type type, string defaultNamespace);

[C++] public: XmlSerializer(Type\* type, String\* defaultNamespace);

[VB] Public Sub New(ByVal type As Type, ByVal defaultNamespace As String)

[JScript] public function XmlSerializer(type : Type, defaultNamespace : String);

## Description

Initializes a new instance of the **System.Xml.Serialization.XmlSerializer** class that can serialize objects of the specified type into XML-document instances, and vice versa. Specifies the default namespace for all the XML elements. The type of the object that this **System.Xml.Serialization.XmlSerializer** can serialize. The default namespace to use for all the XML elements.

XmlSerializer

*Example Syntax:*

WriteXsiType

```
[C#] public XmlSerializer(Type type, Type[] extraTypes);
```

```
[C++] public: XmlSerializer(Type* type, Type* extraTypes[]);
```

```
[VB] Public Sub New(ByVal type As Type, ByVal extraTypes() As Type)
```

```
[JScript] public function XmlSerializer(type : Type, extraTypes : Type[]);
```

## Description

Initializes a new instance of the **System.Xml.Serialization.XmlSerializer** class that can serialize objects of the specified type into XML-document instances, and vice versa. If a property or field returns an array, the *extraTypes* parameter specifies objects that can be inserted into the array.

By default, if a public property or field returns an object, or array of objects, the object types will be automatically serialized. However, if a class contains a field or property that returns an array of type **System.Object**, any object can be inserted into that array. In that case, the

**System.Xml.Serialization.XmlSerializer** must be instructed to expect all of the possible object types that will be inserted into the **System.Object** array. To do this, use the *extraTypes* parameter to specify the extra object types to serialize or deserialize. The type of the object that this **System.Xml.Serialization.XmlSerializer** can serialize. A **System.Type** array of additional object types to serialize. See Remarks below.

XmlSerializer

*Example Syntax:*

WriteXsiType

```
[C#] public XmlSerializer(Type type, XmlAttributeOverrides overrides);
```

```
[C++] public: XmlSerializer(Type* type, XmlAttributeOverrides* overrides);
```

```
[VB] Public Sub New(ByVal type As Type, ByVal overrides As  
XmlAttributeOverrides)
```

```
[JScript] public function XmlSerializer(type : Type, overrides :  
XmlAttributeOverrides);
```

### *Description*

Initializes a new instance of the **System.Xml.Serialization.XmlSerializer** class that can serialize objects of the specified type into XML-document instances, and vice versa. Each object to be serialized can itself contain instances of classes, which this overload can override with other classes.

The *overrides* parameter can be used to control how fields and properties are encoded in XML. These settings override any attributes that already exist on the objects. This can be useful when the source code can not be modified or

multiple encodings are desired for the same classes. The type of the object to serialize. An **System.Xml.Serialization.XmlAttributeOverrides** .

**XmlSerializer**

*Example Syntax:*

**WriteXsiType**

[C#] public XmlSerializer(Type type, XmlRootAttribute root);

[C++] public: XmlSerializer(Type\* type, XmlRootAttribute\* root);

[VB] Public Sub New(ByVal type As Type, ByVal root As XmlRootAttribute)

[JScript] public function XmlSerializer(type : Type, root : XmlRootAttribute);

### *Description*

Initializes a new instance of the **System.Xml.Serialization.XmlSerializer** class that can serialize objects of the specified type into XML-document instances, and vice versa. Specifies the class to use as the XML root element.

The root element of an XML document encloses all the other elements. By default, the object specified by the *type* parameter is serialized as the root element. Properties, such as the XML element name of the root element are taken from the *type* object. However, the *root* parameter allows you to supplant the default object's information by specifying an **System.Xml.Serialization.XmlRootAttribute** ; the object allows you to set a different namespace, element name, and so on. The type of the object that this **System.Xml.Serialization.XmlSerializer** can serialize. An **System.Xml.Serialization.XmlRootAttribute** that represents the XML root element.



XmlSerializer

*Example Syntax:*

WriteXsiType

[C#] public XmlSerializer(Type type, XmlAttributeOverrides overrides, Type[] extraTypes, XmlRootAttribute root, string defaultNamespace);

[C++] public: XmlSerializer(Type\* type, XmlAttributeOverrides\* overrides, Type\* extraTypes[], XmlRootAttribute\* root, String\* defaultNamespace);

[VB] Public Sub New(ByVal type As Type, ByVal overrides As XmlAttributeOverrides, ByVal extraTypes() As Type, ByVal root As XmlRootAttribute, ByVal defaultNamespace As String)

[JScript] public function XmlSerializer(type : Type, overrides : XmlAttributeOverrides, extraTypes : Type[], root : XmlRootAttribute, defaultNamespace : String); Initializes a new instance of the **System.Xml.Serialization.XmlSerializer** class.

### *Description*

Initializes a new instance of the **System.Xml.Serialization.XmlSerializer** class that can serialize objects of the object type into XML document instances, and vice versa. Each object to be serialized can itself contain instances of classes, which this overload can override with other classes. This overload also specifies the default namespace for all the XML elements, and the class to use as the XML root element.

The *overrides* parameter allows for the creation of a serializer that serializes an overriding class. For example, given a DLL it's possible to create a class that

inherits from the `DLL`. To serialize the overriding class, you must use an `System.Xml.Serialization.XmlAttributeOverrides` object when constructing the `System.Xml.Serialization.XmlSerializer`. For more details, see `System.Xml.Serialization.XmlAttributeOverrides`. The type of the object that this `System.Xml.Serialization.XmlSerializer` can serialize. An `System.Xml.Serialization.XmlAttributeOverrides`. See Remarks below. A `System.Type` array of additional object types to serialize. See Remarks below. An `System.Xml.Serialization.XmlRootAttribute` that defines the XML root element properties. The default namespace of all XML elements in the XML-document instance.

WriteXsiType

[C#] public event XmlAttributeEventHandler UnknownAttribute;

[C++] public: \_\_event XmlAttributeEventHandler\* UnknownAttribute;

[VB] Public Event UnknownAttribute As XmlAttributeEventHandler

### *Description*

Occurs when the `System.Xml.Serialization.XmlSerializer` encounters an XML attribute of unknown type during deserialization.

By default, after calling the `System.Xml.Serialization.XmlSerializer.Deserialize(System.IO.Stream)` method, the `System.Xml.Serialization.XmlSerializer` ignores XML attributes of unknown types. However, you can use this event to handle such node types.

WriteXsiType

```

1
2 [C#] public event XmlElementEventHandler UnknownElement;
3 [C++] public: __event XmlElementEventHandler* UnknownElement;
4 [VB] Public Event UnknownElement As XmlElementEventHandler
5

```

### *Description*

Occurs when the **System.Xml.Serialization.XmlSerializer** encounters an XML element of unknown type during deserialization.

By default, after calling the **System.Xml.Serialization.XmlSerializer.Deserialize(System.IO.Stream)** method, the **System.Xml.Serialization.XmlSerializer** ignores XML attributes of unknown types. However, you can use this event to handle such node types.

WriteXsiType

```

13
14
15 [C#] public event XmlNodeEventHandler UnknownNode;
16 [C++] public: __event XmlNodeEventHandler* UnknownNode;
17 [VB] Public Event UnknownNode As XmlNodeEventHandler
18

```

### *Description*

Occurs when the **System.Xml.Serialization.XmlSerializer** encounters an XML node of unknown type during deserialization.

By default, after calling the **System.Xml.Serialization.XmlSerializer.Deserialize(System.IO.Stream)** method, the **System.Xml.Serialization.XmlSerializer** ignores XML nodes of unknown types. However, you can use this event to handle such node types.

## WriteXsiType

```
[C#] public event UnreferencedObjectEventHandler UnreferencedObject;
[C++] public: __event UnreferencedObjectEventHandler* UnreferencedObject;
[VB] Public Event UnreferencedObject As UnreferencedObjectEventHandler
```

### *Description*

## CanDeserialize

```
[C#] public virtual bool CanDeserialize(XmlReader xmlReader);
[C++] public: virtual bool CanDeserialize(XmlReader* xmlReader);
[VB] Overridable Public Function CanDeserialize(ByVal xmlReader As
XmlReader) As Boolean
[JScript] public function CanDeserialize(xmlReader : XmlReader) : Boolean;
```

### *Description*

Gets a value indicating whether this **System.Xml.Serialization.XmlSerializer** can deserialize a specified XML-document instance.

*Return Value:* **true** if an this **System.Xml.Serialization.XmlSerializer** can deserialize the object which the **System.Xml.XmlReader** points to; otherwise, **false** . An **System.Xml.XmlReader** that points to the document to deserialize.

## CreateReader

1 [C#] protected virtual XmlSerializationReader CreateReader();

2 [C++] protected: virtual XmlSerializationReader\* CreateReader();

3 [VB] Overridable Protected Function CreateReader() As XmlSerializationReader

4 [JScript] protected function CreateReader() : XmlSerializationReader;

5  
6  
7 *Description*

8 Returns an object used to read the XML-document instance to be serialized.

9 *Return Value:* An XmlSerializationReader used to read the XML-document  
10 instance.

11 CreateWriter

12  
13 [C#] protected virtual XmlSerializationWriter CreateWriter();

14 [C++] protected: virtual XmlSerializationWriter\* CreateWriter();

15 [VB] Overridable Protected Function CreateWriter() As XmlSerializationWriter

16 [JScript] protected function CreateWriter() : XmlSerializationWriter;

17  
18 *Description*

19  
20 Deserialize

21  
22 [C#] public object Deserialize(Stream stream);

23 [C++] public: Object\* Deserialize(Stream\* stream);

24 [VB] Public Function Deserialize(ByVal stream As Stream) As Object

25 [JScript] public function Deserialize(stream : Stream) : Object; Deserializes an

XML-document instance.

### *Description*

Deserializes the XML-document instance contained by the specified

**System.IO.Stream** .

*Return Value:* The **System.Object** being deserialized.

Deserialization is the process of reading an XML-document instance and constructing an object that is strongly typed to the XML schema of the document.

The **System.IO.Stream** containing the XML-document instance to deserialize.

Deserialize

[C#] public object Deserialize(TextReader textReader);

[C++] public: Object\* Deserialize(TextReader\* textReader);

[VB] Public Function Deserialize(ByVal textReader As TextReader) As Object

[JScript] public function Deserialize(textReader : TextReader) : Object;

### *Description*

Deserializes the XML-document instance contained by the specified

**System.IO.TextReader** .

*Return Value:* The **System.Object** being deserialized.

Deserialization is the process of reading an XML instance document and constructing an object that is strongly typed to the XML schema of the document.

The **System.IO.TextReader** containing the XML-document instance to deserialize.

Deserialize

1  
2 [C#] public object Deserialize(XmlReader xmlReader);

3 [C++] public: Object\* Deserialize(XmlReader\* xmlReader);

4 [VB] Public Function Deserialize(ByVal xmlReader As XmlReader) As Object

5 [JScript] public function Deserialize(xmlReader : XmlReader) : Object;

6  
7 *Description*

8 Deserializes the XML-document instance contained by the specified

9 **System.xml.XmlReader** .

10 *Return Value:* The **System.Object** being deserialized.

11 Deserialization is the process of reading an XML instance document and  
12 constructing an object that is strongly typed to the XML schema of the document.

13 The **System.xml.XmlReader** containing the XML-document instance to  
14 deserialize.

15 Deserialize

16  
17 [C#] protected virtual object Deserialize(XmlSerializationReader reader);

18 [C++] protected: virtual Object\* Deserialize(XmlSerializationReader\* reader);

19 [VB] Overridable Protected Function Deserialize(ByVal reader As  
20 XmlSerializationReader) As Object

21 [JScript] protected function Deserialize(reader : XmlSerializationReader) : Object;

22  
23 *Description*

24  
25 FromMappings

```

1
2 [C#] public static XmlSerializer[] FromMappings(XmlMapping[] mappings);
3 [C++] public: static XmlSerializer* FromMappings(XmlMapping* mappings[]) [];
4 [VB] Public Shared Function FromMappings(ByVal mappings() As XmlMapping)
5 As XmlSerializer()
6 [JScript] public static function FromMappings(mappings : XmlMapping[]) :
7 XmlSerializer[];
8

```

### *Description*

#### FromTypes

```

13 [C#] public static XmlSerializer[] FromTypes(Type[] types);
14 [C++] public: static XmlSerializer* FromTypes(Type* types[]) [];
15 [VB] Public Shared Function FromTypes(ByVal types() As Type) As
16 XmlSerializer()
17 [JScript] public static function FromTypes(types : Type[]) : XmlSerializer[];
18

```

### *Description*

Returns an array of **System.Xml.Serialization.XmlSerializer** objects created from an array of types.

*Return Value:* An array of **System.Xml.Serialization.XmlSerializer** objects.

The **System.Xml.Serialization.XmlSerializer.FromTypes(System.Type[])** method allows you to efficiently create an array of



**System.Xml.Serialization.XmlSerializer** objects for processing an array of **System.Type** objects. An array of **System.Type** objects.

### Serialize

[C#] protected virtual void Serialize(object o, XmlSerializationWriter writer);

[C++] protected: virtual void Serialize(Object\* o, XmlSerializationWriter\* writer);

[VB] Overridable Protected Sub Serialize(ByVal o As Object, ByVal writer As XmlSerializationWriter)

[JScript] protected function Serialize(o : Object, writer : XmlSerializationWriter);

### Description

### Serialize

[C#] public void Serialize(Stream stream, object o);

[C++] public: void Serialize(Stream\* stream, Object\* o);

[VB] Public Sub Serialize(ByVal stream As Stream, ByVal o As Object)

[JScript] public function Serialize(stream : Stream, o : Object);

### Description

Serializes the specified **System.Object** and writes the XML-document instance to a file using the specified **System.IO.Stream** .

The

**System.Xml.Serialization.XmlSerializer.Serialize(System.IO.TextWriter, Syst**

**em.Object)** method converts the public fields and read/write properties of an object into Extensible Markup Language (XML). It does not convert methods, indexers, private fields, or read-only properties. To serialize all of an object's fields and properties, both public and private, use the **System.Runtime.Serialization.Formatters.Binary.BinaryFormatter** . The **System.IO.Stream** used to write the XML-document instance. The **System.Object** to serialize.

#### Serialize

[C#] public void Serialize(TextWriter textWriter, object o);  
 [C++] public: void Serialize(TextWriter\* textWriter, Object\* o);  
 [VB] Public Sub Serialize(ByVal textWriter As TextWriter, ByVal o As Object)  
 [JScript] public function Serialize(textWriter : TextWriter, o : Object); Serializes an object into an XML-document instance.

#### Description

Serializes the specified **System.Object** and writes the XML-document instance to a file using the specified **System.IO.TextWriter** .

The

**System.Xml.Serialization.XmlSerializer.Serialize(System.IO.TextWriter, System.Object)** method converts the public fields and read/write properties of an object into Extensible Markup Language (XML). It does not convert methods, indexers, private fields, or read-only properties. To serialize all of an object's fields and properties, both public and private, use the **System.Runtime.Serialization.Formatters.Binary.BinaryFormatter** . The

**System.IO.TextWriter** used to write the XML-document instance. The **System.Object** to serialize.

Serialize

[C#] public void Serialize(XmlWriter xmlWriter, object o);

[C++] public: void Serialize(XmlWriter\* xmlWriter, Object\* o);

[VB] Public Sub Serialize(ByVal xmlWriter As XmlWriter, ByVal o As Object)

[JScript] public function Serialize(xmlWriter : XmlWriter, o : Object);

### *Description*

Serializes the specified **System.Object** and writes the XML-document instance to a file using the specified **System.Xml.XmlWriter**.

The

**System.Xml.Serialization.XmlSerializer.Serialize(System.IO.TextWriter, System.Object)** method converts the public fields and read/write properties of an object into Extensible Markup Language (XML). It does not convert methods, indexers, private fields, or read-only properties. To serialize all of an object's fields and properties, both public and private, use the

**System.Runtime.Serialization.Formatters.Binary.BinaryFormatter**. The **System.xml.XmlWriter** used to write the XML-document instance. The **System.Object** to serialize.

Serialize

[C#] public void Serialize(Stream stream, object o, XmlSerializerNamespaces namespaces);

```

1 [C++] public: void Serialize(Stream* stream, Object* o,
2 XmlSerializerNamespaces* namespaces);
3 [VB] Public Sub Serialize(ByVal stream As Stream, ByVal o As Object, ByVal
4 namespaces As XmlSerializerNamespaces)
5 [JScript] public function Serialize(stream : Stream, o : Object, namespaces :
6 XmlSerializerNamespaces);

```

### *Description*

Serializes the specified **System.Object** and writes the XML-document instance to a file using the specified **System.IO.Stream** , referencing the specified namespaces.

When the Serialize method is invoked the public fields and read/write properties of an object are converted into Extensible Markup Language (XML). Methods, indexers, private fields, and read-only properties are not serialized. To serialize all fields and properties, both public and private, use the **System.Runtime.Serialization.Formatters.Binary.BinaryFormatter** . The **System.IO.Stream** used to write the XML-document instance. The **System.Object** to serialize. The **System.Xml.Serialization.XmlSerializerNamespaces** referenced by the object.

### Serialize

```

22 [C#] public void Serialize(TextWriter textWriter, object o,
23 XmlSerializerNamespaces namespaces);
24 [C++] public: void Serialize(TextWriter* textWriter, Object* o,
25 XmlSerializerNamespaces* namespaces);

```

```

1 [VB] Public Sub Serialize(ByVal textWriter As TextWriter, ByVal o As Object,
2   ByVal namespaces As XmlSerializerNamespaces)
3 [JScript] public function Serialize(textWriter : TextWriter, o : Object, namespaces
4   : XmlSerializerNamespaces);

```

### *Description*

Serializes the specified **System.Object** and writes the XML-document instance to a file using the specified **System.IO.TextWriter** , referencing the specified namespaces.

When the **System.Xml.Serialization.XmlSerializer.Serialize(System.IO.TextWriter, System.Object)** method is invoked the public fields and read/write properties of an object are converted into Extensible Markup Language (XML). Methods, indexers, private fields, and read-only properties are not serialized. To serialize all fields and properties, both public and private, use the **System.Runtime.Serialization.Formatters.Binary.BinaryFormatter** . The **System.IO.TextWriter** used to write the XML-document instance. The **System.Object** to serialize. The **System.Xml.Serialization.XmlSerializerNamespaces** containing namespaces for the generated XML document.

### *Serialize*

```

23 [C#] public void Serialize(XmlWriter xmlWriter, object o,
24   XmlSerializerNamespaces namespaces);
25 [C++] public: void Serialize(XmlWriter* xmlWriter, Object* o,

```

1 XmlSerializerNamespaces\* namespaces);

2 [VB] Public Sub Serialize(ByVal xmlWriter As XmlWriter, ByVal o As Object,  
3 ByVal namespaces As XmlSerializerNamespaces)

4 [JScript] public function Serialize(xmlWriter : XmlWriter, o : Object, namespaces  
5 : XmlSerializerNamespaces);

### 6 7 *Description*

8 Serializes the specified **System.Object** and writes the XML-document  
9 instance to a file using the specified **System.Xml.XmlWriter** , referencing the  
10 specified namespaces.

11 When the  
12 **System.Xml.Serialization.XmlSerializer.Serialize(System.IO.TextWriter, System.Object)** method is invoked the public fields and read/write properties of an  
13 object are converted into Extensible Markup Language (XML). Methods,  
14 indexers, private fields, and read-only properties are not serialized. To serialize all  
15 fields and properties, both public and private, use the  
16 **System.Runtime.Serialization.Formatters.Binary.BinaryFormatter** . The  
17 **System.xml.XmlWriter** used to write the XML-document instance. The  
18 **System.Object** to serialize. The  
19 **System.Xml.Serialization.XmlSerializerNamespaces** referenced by the object.

20  
21 XmlSerializerNamespaces class (System.Xml.Serialization)

22 ToString

### 23 24 25 *Description*

1 Contains the XML namespaces and prefixes that the  
2 **System.Xml.Serialization.XmlSerializer** uses to generate qualified names in an  
3 XML-document instance.

4 The **System.Xml.Serialization.XmlSerializerNamespaces** contains a  
5 collection of XML namespaces, each with an associated prefix. The  
6 **System.Xml.Serialization.XmlSerializer** uses an instance of the  
7 **System.Xml.Serialization.XmlSerializerNamespaces** class to create qualified  
8 names in an XML-document instance.

9 XmlSerializerNamespaces

10 *Example Syntax:*

11 ToString

12  
13 [C#] public XmlSerializerNamespaces();

14 [C++] public: XmlSerializerNamespaces();

15 [VB] Public Sub New()

16 [JScript] public function XmlSerializerNamespaces(); Initializes a new instance of  
17 the **System.Xml.Serialization.XmlSerializerNamespaces** class.

18  
19 *Description*

20 Initializes a new instance of the  
21 **System.Xml.Serialization.XmlSerializerNamespaces** class.

22 XmlSerializerNamespaces

23 *Example Syntax:*

24 ToString

```

1
2 [C#] public XmlSerializerNamespaces(XmlQualifiedName[] namespaces);
3 [C++] public: XmlSerializerNamespaces(XmlQualifiedName* namespaces[]);
4 [VB] Public Sub New(ByVal namespaces() As XmlQualifiedName)
5 [JScript] public function XmlSerializerNamespaces(namespaces :
6 XmlQualifiedName[]);

```

### *Description*

Initializes a new instance of the **System.Xml.Serialization.XmlSerializerNamespaces** class, using the specified array of **System.Xml.XmlQualifiedName** objects to create a collection of prefix-namespace pairs. An array of **System.Xml.XmlQualifiedName** objects.

Count

ToString

```

16 [C#] public int Count {get;}
17 [C++] public: __property int get_Count();
18 [VB] Public ReadOnly Property Count As Integer
19 [JScript] public function get Count() : int;

```

### *Description*

Gets the number of prefix-namespace pairs in the collection.

Add

```

25 [C#] public void Add(string prefix, string ns);

```



1 [C++] public: void Add(String\* prefix, String\* ns);

2 [VB] Public Sub Add(ByVal prefix As String, ByVal ns As String)

3 [JScript] public function Add(prefix : String, ns : String);

4  
5 *Description*

6 Adds a prefix-namespace pair to an

7 **System.Xml.Serialization.XmlSerializerNamespaces** object.

8 If you want the **System.Xml.Serialization.XmlSerializer** to qualify the  
9 element and attribute names in an XML document, you must

10 **System.Xml.Serialization.XmlSerializerNamespaces.Add(System.String, System.String)** the prefix-namespace pairs to an

11 **System.Xml.Serialization.XmlSerializerNamespaces** object. The prefix  
12 associated with an XML namespace. An XML namespace.

13  
14 *ToArray*

15  
16 [C#] public XmlQualifiedName[] ToArray();

17 [C++] public: XmlQualifiedName\* ToArray() [];

18 [VB] Public Function ToArray() As XmlQualifiedName()

19 [JScript] public function ToArray() : XmlQualifiedName[];

20  
21 *Description*

22 Gets the array of prefix-namespace pairs in an

23 **System.Xml.Serialization.XmlSerializerNamespaces** object.

24 *Return Value:* An array of **System.Xml.XmlQualifiedName** objects that are used  
25 as qualified names in an XML-document instance.

XmlTextAttribute class (System.Xml.Serialization)

ToString

### *Description*

Indicates to the **System.Xml.Serialization.XmlSerializer** that the member should be treated as XML text when the containing class is serialized or deserialized.

The **System.Xml.Serialization.XmlTextAttribute** belongs to a family of attributes that controls how the **System.Xml.Serialization.XmlSerializer** serializes and deserializes an object (through its **System.Xml.Serialization.XmlSerializer.Serialize(System.IO.TextWriter,System.Object)** and **System.Xml.Serialization.XmlSerializer.Deserialize(System.IO.Stream)** methods). See the **System.Xml.Serialization.XmlSerializer** class for a complete list of similar attributes.

XmlTextAttribute

*Example Syntax:*

ToString

[C#] public XmlTextAttribute();

[C++] public: XmlTextAttribute();

[VB] Public Sub New()

[JScript] public function XmlTextAttribute(); Initializes a new instance of the

**System.Xml.Serialization.XmlTextAttribute** class.

## Description

Initializes a new instance of the

**System.Xml.Serialization.XmlTextAttribute** class.

You can override the way that the

**System.Xml.Serialization.XmlSerializer** serializes a public field or public read/write property by creating an an **System.Xml.Serialization.XmlAttributes** object, and setting its **System.Xml.Serialization.XmlAttributes.XmlText** property to an **System.Xml.Serialization.XmlTextAttribute** object. For more details, see the **System.Xml.Serialization.XmlAttributeOverrides** class.

XmlTextAttribute

*Example Syntax:*

ToString

[C#] public XmlTextAttribute(Type type);

[C++] public: XmlTextAttribute(Type\* type);

[VB] Public Sub New(ByVal type As Type)

[JScript] public function XmlTextAttribute(type : Type);

## Description

Initializes a new instance of the

**System.Xml.Serialization.XmlTextAttribute** class.

Type

ToString

```

1
2 [C#] public Type Type {get; set;}
3 [C++] public: __property Type* get_Type();public: __property void
4 set_Type(Type*);
5 [VB] Public Property Type As Type
6 [JScript] public function get Type() : Type;public function set Type(Type);

```

### Description

Gets or sets the type of the member.

TypeId

XmlAttribute class (System.Xml.Serialization)

ToString

### Description

Represents a complex type XML element.

The **System.Xml.Serialization.XmlTypeAttribute** belongs to a family of attributes that controls how the **System.Xml.Serialization.XmlSerializer** serializes or deserializes an object. For a complete list of similar attributes, see the **System.Xml.Serialization.XmlSerializer** class.

XmlAttribute

*Example Syntax:*

ToString

```

25 [C#] public XmlTypeAttribute();

```

[C++] public: XmlTypeAttribute();  
 [VB] Public Sub New()  
 [JScript] public function XmlTypeAttribute(); Initializes a new instance of the **System.Xml.Serialization.XmlTypeAttribute** class.

### *Description*

Initializes a new instance of the **System.Xml.Serialization.XmlTypeAttribute** class.

XmlTypeAttribute

*Example Syntax:*

ToString

[C#] public XmlTypeAttribute(string typeName);  
 [C++] public: XmlTypeAttribute(String\* typeName);  
 [VB] Public Sub New(ByVal typeName As String)  
 [JScript] public function XmlTypeAttribute(typeName : String);

### *Description*

Initializes a new instance of the **System.Xml.Serialization.XmlTypeAttribute** class, specifying the name of the XML type. The name of the XML type that the **System.Xml.Serialization.XmlSerializer** generates when it serializes the class instance (and recognizes when it deserializes the class instance).

IncludeInSchema

ToString

1  
2 [C#] public bool IncludeInSchema {get; set;}

3 [C++] public: \_\_property bool get\_IncludeInSchema();public: \_\_property void  
4 set\_IncludeInSchema(bool);

5 [VB] Public Property IncludeInSchema As Boolean

6 [JScript] public function get IncludeInSchema() : Boolean;public function set  
7 IncludeInSchema(Boolean);

8  
9 *Description*

10 Gets or sets a value indicating whether to include the type in XML schema  
11 documents.

12 Apply the **System.Xml.Serialization.XmlTypeAttribute** to a class to  
13 specify the XML type's namespace, the XML type name, and whether to include  
14 the type in the XML schema document. To see the results of setting the  
15 **System.Xml.Serialization.XmlTypeAttribute** class's properties, compile your  
16 application as an executable or DLL, and pass the resulting file to the XML  
17 Schema Definition tool (XSD.exe). The tool writes the schema--including the type  
18 definition.

19 Namespace

20 ToString

21  
22 [C#] public string Namespace {get; set;}

23 [C++] public: \_\_property String\* get\_Namespace();public: \_\_property void  
24 set\_Namespace(String\*);

25 [VB] Public Property Namespace As String

1 [JScript] public function get Namespace() : String;public function set

2 Namespace(String);

3  
4 *Description*

5 Gets or sets the namespace of the XML type.

6 TypeId

7 TypeName

8 ToString

9  
10  
11 *Description*

12 Gets or sets the name of the XML type.

13 Apply the **System.Xml.Serialization.XmlTypeAttribute** to a class to  
14 specify the XML type's namespace, the XML type name, and whether to include  
15 the type in the XML schema document. To see the results of setting the  
16 **System.Xml.Serialization.XmlTypeAttribute** class's properties, compile your  
17 application as an executable or DLL, and pass the resulting file to the XML  
18 Schema Definition tool (XSD.exe). The tool writes the schema--including the type  
19 definition.

20 XmlTypeMapping class (System.Xml.Serialization)

21 ToString

22  
23  
24 *Description*

```

1      ElementName
2
3
4      [C#] public string ElementName {get;}
5      [C++] public: __property String* get_ElementName();
6      [VB] Public ReadOnly Property ElementName As String
7      [JScript] public function get ElementName() : String;
8
9      Description
10
11      Namespace
12      ToString
13
14      [C#] public string Namespace {get;}
15      [C++] public: __property String* get_Namespace();
16      [VB] Public ReadOnly Property Namespace As String
17      [JScript] public function get Namespace() : String;
18
19      Description
20
21      TypeFullName
22      ToString
23
24      [C#] public string TypeFullName {get;}
25      [C++] public: __property String* get_TypeFullName();

```



1 [VB] Public ReadOnly Property TypeFullName As String

2 [JScript] public function get TypeFullName() : String;

3  
4 *Description*

5  
6       TypeName

7       ToString

8  
9 [C#] public string TypeName {get;}

10 [C++] public: \_\_property String\* get\_TypeName();

11 [VB] Public ReadOnly Property TypeName As Strin

12  
13 **System.Xml.XPath**

14       This namespace contains the XPath parser and evaluation engine.

15  
16 *Description*

17       This namespace contains the XPath parser and evaluation engine.

18       IXPathNavigable interface (System.Xml.XPath)

19  
20  
21 *Description*

22       Provides an accessor to the **System.Xml.XPath.XPathNavigator** .

23       Methods:

24       CreateNavigator

1  
2 [C#] XPathNavigator CreateNavigator();

3 [C++] XPathNavigator\* CreateNavigator();

4 [VB] Function CreateNavigator() As XPathNavigator

5 [JScript] function CreateNavigator() : XPathNavigator;

6  
7 *Description*

8 Creates a new **System.Xml.XPath.XPathNavigator** for this  
9 **IXPathNavigable** interface.

10 *Return Value:* An **XPathNavigator** object.

11 If the **CreateNavigator** method is called from an **System.Xml.XmlNode**  
12 object, the **XPathNavigator** is initially positioned on the node from which the  
13 method was called. If **CreateNavigator** is called from an  
14 **System.Xml.XPath.XPathDocument** object, the **XPathNavigator** is positioned  
15 on the root of the document.

16 XmlCaseOrder enumeration (System.Xml.XPath)

17 CreateNavigator

18  
19  
20 *Description*

21 Specifies the sort order for upper and lower case letters.

22 CreateNavigator

23  
24 [C#] public const XmlCaseOrder LowerFirst;

25 [C++] public: const XmlCaseOrder LowerFirst;

[VB] Public Const LowerFirst As XmlCaseOrder

[JScript] public var LowerFirst : XmlCaseOrder;

*Description*

Lower case letters are sorted before upper case letters.

CreateNavigator

[C#] public const XmlCaseOrder None;

[C++] public: const XmlCaseOrder None;

[VB] Public Const None As XmlCaseOrder

[JScript] public var None : XmlCaseOrder;

*Description*

Ignore the case.

CreateNavigator

[C#] public const XmlCaseOrder UpperFirst;

[C++] public: const XmlCaseOrder UpperFirst;

[VB] Public Const UpperFirst As XmlCaseOrder

[JScript] public var UpperFirst : XmlCaseOrder;

*Description*

Upper case letters are sorted before lower case letters.

XmlDataType enumeration (System.Xml.XPath)

ToString

MSI-865US.APP  
509-324-9256  
lee@hayes.plc

*Description*

Specifies the data type used to determine sort order.

ToString

[C#] public const XmlDataType Number;

[C++] public: const XmlDataType Number;

[VB] Public Const Number As XmlDataType

[JScript] public var Number : XmlDataType;

*Description*

Values are sorted numerically.

ToString

[C#] public const XmlDataType Text;

[C++] public: const XmlDataType Text;

[VB] Public Const Text As XmlDataType

[JScript] public var Text : XmlDataType;

*Description*

Values are sorted alphabetically.

XmlSortOrder enumeration (System.Xml.XPath)

ToString

*Description*

Specifies the sort order.

ToString

[C#] public const XmlSortOrder Ascending;

[C++] public: const XmlSortOrder Ascending;

[VB] Public Const Ascending As XmlSortOrder

[JScript] public var Ascending : XmlSortOrder;

*Description*

Nodes are sorted in ascending order. For example, if the numbers 1,2,3,4 were being sorted in an ascending order, they would appear as 1,2,3,4.

ToString

[C#] public const XmlSortOrder Descending;

[C++] public: const XmlSortOrder Descending;

[VB] Public Const Descending As XmlSortOrder

[JScript] public var Descending : XmlSortOrder;

*Description*

Nodes are sorted in descending order. For example, if the numbers 1,2,3,4 were being sorted in an descending order, they would appear as, 4,3,2,1.

XPathDocument class (System.Xml.XPath)

ToString

*Description*

Provides a fast and performant read-only cache for XML document processing using XSLT.

This class is optimized for XSLT processing and the XPath data model. It does not maintain node identity nor does it do all of the rule checking required by the W3C DOM.

Constructors:

XPathDocument

*Example Syntax:*

ToString

[C#] public XPathDocument(Stream stream);

[C++] public: XPathDocument(Stream\* stream);

[VB] Public Sub New(ByVal stream As Stream)

[JScript] public function XPathDocument(stream : Stream);

*Description*

Initializes a new instance of the **XPathDocument** class. The stream containing the data to load.

XPathDocument

*Example Syntax:*

ToString

```

1
2 [C#] public XPathDocument(string uri);
3 [C++] public: XPathDocument(String* uri);
4 [VB] Public Sub New(ByVal uri As String)
5 [JScript] public function XPathDocument(uri : String);
6

```

#### *Description*

Initializes a new instance of the **XPathDocument** class. A URI that specifies a file containing the data to load.

XPathDocument

*Example Syntax:*

ToString

```

13
14 [C#] public XPathDocument(TextReader reader);
15 [C++] public: XPathDocument(TextReader* reader);
16 [VB] Public Sub New(ByVal reader As TextReader)
17 [JScript] public function XPathDocument(reader : TextReader);
18

```

#### *Description*

Initializes a new instance of the **XPathDocument** class. A **System.IO.TextReader** containing the data to load.

XPathDocument

*Example Syntax:*

ToString

```

1
2 [C#] public XPathDocument(XmlReader reader);
3 [C++] public: XPathDocument(XmlReader* reader);
4 [VB] Public Sub New(ByVal reader As XmlReader)
5 [JScript] public function XPathDocument(reader : XmlReader);
6

```

### *Description*

Initializes a new instance of the **XPathDocument** class.

If the **XmlReader** is positioned on a node, the data is loaded from the current position of the **XmlReader** through all its siblings. This enables you to load only a portion of a document. An **System.Xml.XmlReader** containing the data to load.

**XPathDocument**

*Example Syntax:*

**ToString**

```

17 [C#] public XPathDocument(string uri, XmlSpace space);
18 [C++] public: XPathDocument(String* uri, XmlSpace space);
19 [VB] Public Sub New(ByVal uri As String, ByVal space As XmlSpace)
20 [JScript] public function XPathDocument(uri : String, space : XmlSpace);
21

```

### *Description*

Initializes a new instance of the **XPathDocument** class. A URI that specifies a file containing the data to load. An **System.Xml.XmlSpace** value indicating indicating whether to preserve whitespace. Setting this to



1 XmlSpace.Default preserves only significant whitespace; XmlSpace.Preserve  
2 preserves all whitespace.

3 XPathDocument

4 *Example Syntax:*

5 ToString

6  
7 [C#] public XPathDocument(XmlReader reader, XmlSpace space);

8 [C++] public: XPathDocument(XmlReader\* reader, XmlSpace space);

9 [VB] Public Sub New(ByVal reader As XmlReader, ByVal space As XmlSpace)

10 [JScript] public function XPathDocument(reader : XmlReader, space : XmlSpace);

11 Initializes a new instance of the **XPathDocument** class.

12  
13 *Description*

14 Initializes a new instance of the **XPathDocument** class.

15 If the **XmlReader** is positioned on a node, the data is loaded from the  
16 current position of the **XmlReader** through all its siblings. This enables you to  
17 load only a portion of a document. An **System.Xml.XmlReader** containing the  
18 data to load. An **System.Xml.XmlSpace** value indicating whether to preserve  
19 whitespace. Setting this to XmlSpace.Default preserves only significant  
20 whitespace; XmlSpace.Preserve preserves all whitespace.

21 CreateNavigator

22  
23 [C#] public XPathNavigator CreateNavigator();

24 [C++] public: \_\_sealed XPathNavigator\* CreateNavigator();

25 [VB] NotOverridable Public Function CreateNavigator() As XPathNavigator

1 [JScript] public function CreateNavigator() : XPathNavigator;

2  
3 *Description*

4 Creates an **System.Xml.XPath.XPathNavigator** for navigating this  
5 document.

6 *Return Value:* An **System.Xml.XPath.XPathNavigator** object.

7 The **XPathNavigator** is positioned at the root of the document. This  
8 method is part of the **System.Xml.XPath.IXPathNavigable** interface.

9 XPathException class (System.Xml.XPath)

10 ToString

11  
12  
13 *Description*

14 Represents the exception that is thrown when there is error processing an  
15 XPath expression.

16 XPathException

17 *Example Syntax:*

18 ToString

19  
20 [C#] public XPathException(string res);

21 [C++] public: XPathException(String\* res);

22 [VB] Public Sub New(ByVal res As String)

23 [JScript] public function XPathException(res : String);

24  
25 *Description*

XPathException

*Example Syntax:*

ToString

[C#] public XPathException(SerializationInfo info, StreamingContext context);

[C++] public: XPathException(SerializationInfo\* info, StreamingContext  
context);

[VB] Public Sub New(ByVal info As SerializationInfo, ByVal context As  
StreamingContext)

[JScript] public function XPathException(info : SerializationInfo, context :  
StreamingContext);

*Description*

XPathException

*Example Syntax:*

ToString

[C#] public XPathException(string message, Exception innerException);

[C++] public: XPathException(String\* message, Exception\* innerException);

[VB] Public Sub New(ByVal message As String, ByVal innerException As  
Exception)

[JScript] public function XPathException(message : String, innerException :  
Exception);

XPathException

*Example Syntax:*

ToString

[C#] public XPathException(string res, string arg);

[C++] public: XPathException(String\* res, String\* arg);

[VB] Public Sub New(ByVal res As String, ByVal arg As String)

[JScript] public function XPathException(res : String, arg : String);

*Description*

XPathException

*Example Syntax:*

ToString

[C#] public XPathException(string res, string[] args);

[C++] public: XPathException(String\* res, String\* args \_\_gc[]);

[VB] Public Sub New(ByVal res As String, ByVal args() As String)

[JScript] public function XPathException(res : String, args : String[]);

*Description*

XPathException

*Example Syntax:*

ToString

```

1
2 [C#] public XPathException(string res, string arg1, string arg2);
3 [C++] public: XPathException(String* res, String* arg1, String* arg2);
4 [VB] Public Sub New(ByVal res As String, ByVal arg1 As String, ByVal arg2 As
5 String)
6 [JScript] public function XPathException(res : String, arg1 : String, arg2 : String);
7

```

### Description

#### Properties:

HelpLink

HResult

InnerException

Message

ToString

### Description

Source

StackTrace

TargetSite

GetObjectData

```

24 [C#] public override void GetObjectData(SerializationInfo info, StreamingContext
25 context);

```

```

1 [C++] public: void GetObjectData(SerializationInfo* info, StreamingContext
2 context);
3 [VB] Overrides Public Sub GetObjectData(ByVal info As SerializationInfo,
4 ByVal context As StreamingContext)
5 [JScript] public override function GetObjectData(info : SerializationInfo, context :
6 StreamingContext);

```

#### *Description*

XPathExpression class (System.Xml.XPath)  
ToString

#### *Description*

Encapsulates a compiled XPath expression. This class is returned as a result of a call to **System.Xml.XPath.XPathNavigator.Compile(System.String)** and is used by the **System.Xml.XPath.XPathNavigator.Select(System.Xml.XPath.XPathExpression)** , **System.Xml.XPath.XPathNavigator.Evaluate(System.Xml.XPath.XPathExpression)** and **System.Xml.XPath.XPathNavigator.Matches(System.Xml.XPath.XPathExpression)** methods.

Expression  
ToString

1  
2 [C#] public abstract string Expression {get;}

3 [C++] public: \_\_property virtual String\* get\_Expression() = 0;

4 [VB] MustOverride Public ReadOnly Property Expression As String

5 [JScript] public abstract function get Expression() : String;

6  
7 *Description*

8 When overridden in a derived class, gets a string representation of the the

9 **XPathExpression** .

10  
11 *Description*

12 Gets a string representation of the **XPathExpression** .

13 This could be used for diagnostic purposes or as input to another  
14 expression.

15 Return Type

16 ToString

17  
18 [C#] public abstract XPathResultType Return Type {get;}

19 [C++] public: \_\_property virtual XPathResultType get\_ReturnType() = 0;

20 [VB] MustOverride Public ReadOnly Property Return Type As XPathResultType

21 [JScript] public abstract function get Return Type() : XPathResultType;

22  
23 *Description*

24 Gets the result type of the **XPathExpression** as defined by the W3C XPath  
25 specification.

## AddSort

```
[C#] public abstract void AddSort(object expr, IComparer comparer);
[C++] public: virtual void AddSort(Object* expr, IComparer* comparer) = 0;
[VB] MustOverride Public Sub AddSort(ByVal expr As Object, ByVal comparer
As IComparer)
[JavaScript] public abstract function AddSort(expr : Object, comparer : IComparer);
```

Sorts the nodes selected by the **XPathExpression** .

### *Description*

Sorts the nodes selected by the **XPathExpression** , according to the **System.Collections.IComparer** interface.

This method enables users to sort objects by their data type rather than by string or number. The **IComparer** interface provides an implementation of the **System.Collections.IComparer.Compare(System.Object, System.Object)** method that supports sorting on user defined classes. An expression representing the sort key. This can be a string or an **XPathExpression** object. The result of this expression is converted to a string, according to the XPath specification, for comparison. In an XSLT stylesheet, if xsl:sort is used, but no select expression is specified, then string(.) is used by default. A class derived from the **IComparer** interface to use for the data type comparison.

## AddSort

```
[C#] public abstract void AddSort(object expr, XmlSortOrder order,
XmlCaseOrder caseOrder, string lang, XmlDataType dataType);
```



```

1 [C++] public: virtual void AddSort(Object* expr, XmlSortOrder order,
2 XmlCaseOrder caseOrder, String* lang, XmlDataType dataType) = 0;
3 [VB] MustOverride Public Sub AddSort(ByVal expr As Object, ByVal order As
4 XmlSortOrder, ByVal caseOrder As XmlCaseOrder, ByVal lang As String, ByVal
5 dataType As XmlDataType)
6 [JScript] public abstract function AddSort(expr : Object, order : XmlSortOrder,
7 caseOrder : XmlCaseOrder, lang : String, dataType : XmlDataType);
8

```

### Description

Sorts the nodes selected by the **XPathExpression** according to the supplied parameters.

The order in which the sorts are added provides the sort key order. An expression representing the sort key. This can be a string or an **XPathExpression** object. The result of this expression is converted to a string, according to the XPath specification, for comparison. In an XSLT stylesheet, if **xsl:sort** is used, but no select expression is specified, then **string(.)** is used by default. A

**System.Xml.XPath.XmlSortOrder** value indicating the sort order. A

**System.Xml.XPath.XmlCaseOrder** value indicating how to sort upper/lower case letters. This is language dependent, providing a *lang* parameter is supplied.

The language to use for comparison. Uses the **System.Globalization.CultureInfo** class that can be passed to the **String.Compare** method for the language types, for example, "us-en" for US English. If an empty string is specified, the system environment is used to determine the **CultureInfo**.

**System.Xml.XPath.XmlDataType** indicating sort order for data type.

Clone

```

1
2 [C#] public abstract XPathExpression Clone();
3 [C++] public: virtual XPathExpression* Clone() = 0;
4 [VB] MustOverride Public Function Clone() As XPathExpression
5 [JScript] public abstract function Clone() : XPathExpression;
6

```

### Description

Clones the **XPathExpression** .

*Return Value:* A new **XPathExpression** object.

### SetContext

```

12 [C#] public abstract void SetContext(XmlNamespaceManager nsManager);
13 [C++] public: virtual void SetContext(XmlNamespaceManager* nsManager) = 0;
14 [VB] MustOverride Public Sub SetContext(ByVal nsManager As
15 XmlNamespaceManager)
16 [JScript] public abstract function SetContext(nsManager :
17 XmlNamespaceManager);
18

```

### Description

Specifies the **System.Xml.XmlNamespaceManager** to use for resolving namespaces.

Namespace resolution is supported using the **System.Xml.XmlNamespaceManager** which stores prefix and namespace URI mappings. If the **XPathExpression** requires namespace resolution, the prefix and namespace URI pair must be added to the **XmlNamespaceManager** , and the

**SetContext** method must be called to specify the **XmlNamespaceManager** to use for namespace resolution. The **XmlNamespaceManager** object used for resolving namespaces.

XPathNavigator class (System.Xml.XPath)

ToString

### *Description*

Reads data from any data store using a cursor model.

**XPathNavigator** is based on the XPath data model and provides the methods required to implement XPath queries over any data store. To implement an **XPathNavigator** over an XML document or a **System.Data.DataSet** (via the **System.Xml.XmlDataDocument** class), use the **CreateNavigator** method available in the **System.Xml.XmlNode** and **System.Xml.XPath.XPathDocument** classes, or in the **System.Xml.XPath.IXPathNavigable** interface. Additionally, you can create your own implementation of **XPathNavigator** over other data stores such as file systems or databases.

XPathNavigator

*Example Syntax:*

ToString

[C#] protected XPathNavigator();

[C++] protected: XPathNavigator();

[VB] Protected Sub New()

[JScript] protected function XPathNavigator();

BaseURI

ToString

[C#] public abstract string BaseURI {get;}

[C++] public: \_\_property virtual String\* get\_BaseURI() = 0;

[VB] MustOverride Public ReadOnly Property BaseURI As String

[JScript] public abstract function get BaseURI() : String;

### *Description*

When overridden in a derived class, gets the base URI for the current node.

A networked XML document is comprised of chunks of data, aggregated using various W3C standard inclusion mechanisms, and therefore contains nodes that come from different places. DTD entities are an example of this, however, this is not limited to DTDs. The **BaseURI** property tells you where these nodes originate from.

HasAttributes

ToString

[C#] public abstract bool HasAttributes {get;}

[C++] public: \_\_property virtual bool get\_HasAttributes() = 0;

[VB] MustOverride Public ReadOnly Property HasAttributes As Boolean

[JScript] public abstract function get HasAttributes() : Boolean;

### *Description*

When overridden in a derived class, gets a value indicating whether the element node has any attributes.

The **XPathNavigator** should be positioned on an element node before calling this method.

HasChildren

ToString

[C#] public abstract bool HasChildren {get;}

[C++] public: \_\_property virtual bool get\_HasChildren() = 0;

[VB] MustOverride Public ReadOnly Property HasChildren As Boolean

[JScript] public abstract function get HasChildren() : Boolean;

### *Description*

When overridden in a derived class, gets a value indicating whether the current node has child nodes.

The following table lists the **System.Xml.XPath.XPathNodeType** and what type of child nodes they may have, if any.

IsEmptyElement

ToString

[C#] public abstract bool IsEmptyElement {get;}

[C++] public: \_\_property virtual bool get\_IsEmptyElement() = 0;

[VB] MustOverride Public ReadOnly Property IsEmptyElement As Boolean

[JScript] public abstract function get IsEmptyElement() : Boolean;

## Description

When overridden in a derived class, gets a value indicating whether the current node is an empty element (for example, ).

This property enables you to determine the difference between the following: ( **IsEmptyElement** is **true** ).

LocalName

ToString

```
[C#] public abstract string LocalName {get;}
```

```
[C++] public: __property virtual String* get_LocalName() = 0;
```

```
[VB] MustOverride Public ReadOnly Property LocalName As String
```

```
[JScript] public abstract function get LocalName() : String;
```

## Description

When overridden in a derived class, gets the name of the current node without the namespace prefix.

Name

ToString

```
[C#] public abstract string Name {get;}
```

```
[C++] public: __property virtual String* get_Name() = 0;
```

```
[VB] MustOverride Public ReadOnly Property Name As String
```

```
[JScript] public abstract function get Name() : String;
```

1  
2 *Description*

3       When overridden in a derived class, gets the qualified name of the current  
4 node.

5       NamespaceURI

6       ToString

7  
8 [C#] public abstract string NamespaceURI {get;}

9 [C++] public: \_\_property virtual String\* get\_NamespaceURI() = 0;

10 [VB] MustOverride Public ReadOnly Property NamespaceURI As String

11 [JScript] public abstract function get NamespaceURI() : String;

12  
13 *Description*

14       When overridden in a derived class, gets the namespace URI (as defined in  
15 the W3C Namespace Specification) of the current node.

16       Only element and attribute nodes can have a namespace URI.

17       NameTable

18       ToString

19  
20 [C#] public abstract XmlNameTable NameTable {get;}

21 [C++] public: \_\_property virtual XmlNameTable\* get\_NameTable() = 0;

22 [VB] MustOverride Public ReadOnly Property NameTable As XmlNameTable

23 [JScript] public abstract function get NameTable() : XmlNameTable;

24  
25 *Description*

When overridden in a derived class, gets the **System.Xml.XmlNameTable** associated with this implementation.

The **XmlNameTable** stores atomized strings of any **System.Xml.XPath.XPathNavigator.LocalName** , **System.Xml.XPath.XPathNavigator.NamespaceURI** , and **System.Xml.XPath.XPathNavigator.Prefix** used by **XPathNavigator** . This means that when the same name is returned multiple times (like "book") then the same **String** object is returned for that name. This makes it possible to write efficient code that does object comparisons on these strings instead of expensive string comparisons.

**NodeType**

**ToString**

[C#] public abstract XPathNodeType NodeType {get;}

[C++] public: \_\_property virtual XPathNodeType get\_NodeType() = 0;

[VB] MustOverride Public ReadOnly Property NodeType As XPathNodeType

[JScript] public abstract function get NodeType() : XPathNodeType;

### *Description*

When overridden in a derived class, gets the type of the current node.

**Prefix**

**ToString**

[C#] public abstract string Prefix {get;}

[C++] public: \_\_property virtual String\* get\_Prefix() = 0;



[VB] MustOverride Public ReadOnly Property Prefix As String

[JScript] public abstract function get Prefix() : String;

#### *Description*

When overridden in a derived class, gets the atomized string of the prefix associated with the current node.

Value

ToString

[C#] public abstract string Value {get;}

[C++] public: \_\_property virtual String\* get\_Value() = 0;

[VB] MustOverride Public ReadOnly Property Value As String

[JScript] public abstract function get Value() : String;

#### *Description*

When overridden in a derived class, gets the text value of the current node.

XmlLang

ToString

[C#] public abstract string XmlLang {get;}

[C++] public: \_\_property virtual String\* get\_XmlLang() = 0;

[VB] MustOverride Public ReadOnly Property XmlLang As String

[JScript] public abstract function get XmlLang() : String;

#### *Description*

When overridden in a derived class, gets the **xml:lang** scope for the current node.

Using the XML text below, if the navigator were positioned on the **name** element, the user can call this property to find out that the element is in the scope of a US English **xml:lang** attribute.

#### Clone

[C#] public abstract XPathNavigator Clone();

[C++] public: virtual XPathNavigator\* Clone() = 0;

[VB] MustOverride Public Function Clone() As XPathNavigator

[JScript] public abstract function Clone() : XPathNavigator;

#### Description

When overridden in a derived class, creates a new **XPathNavigator** positioned at the same node as this **XPathNavigator**.

*Return Value:* A new **XPathNavigator** object.

The cloned **XPathNavigator** is not affected by subsequent changes to the current **XPathNavigator**.

#### ComparePosition

[C#] public virtual XmlNodeOrder ComparePosition(XPathNavigator nav);

[C++] public: virtual XmlNodeOrder ComparePosition(XPathNavigator\* nav);

[VB] Overridable Public Function ComparePosition(ByVal nav As XPathNavigator) As XmlNodeOrder

[JScript] public function ComparePosition(nav : XPathNavigator) :

XmlNodeOrder;

### Description

Compares the position of the current navigator with the position of the specified **XPathNavigator** .

*Return Value:* An **System.Xml.XmlNodeOrder** value representing the comparative position of the two navigators. The following table describes the **XmlNodeOrder** enumeration.

The method behavior is dependent on the node type the **XPathNavigator** is currently positioned on. When comparing nodes in the XML document, the following rules apply: Element nodes - These nodes are returned in document order from the source document. The **XPathNavigator** to compare against.

### Compile

[C#] public virtual XPathExpression Compile(string xpath);

[C++] public: virtual XPathExpression\* Compile(String\* xpath);

[VB] Overridable Public Function Compile(ByVal xpath As String) As XPathExpression

[JScript] public function Compile(xpath : String) : XPathExpression;

### Description

Compiles a string representing an XPath expression and returns an **System.Xml.XPath.XPathExpression** .

*Return Value:* An **XPathExpression** object representing the XPath expression.

An XPath expression is evaluated to yield one of the following return types:

- Node-set - an unordered collection of nodes without duplicates
- Boolean - **true** or **false**
- Number - a floating-point number
- String - a sequence of UCS characters

Expressions that return a node-set can be used in the **System.Xml.XPath.XPathNavigator.Select(System.Xml.XPath.XPathExpression)** method. Expressions that return a boolean, number, or string can be used in the **System.Xml.XPath.XPathNavigator.Evaluate(System.Xml.XPath.XPathExpression)** method. The rules on valid expressions for the **System.Xml.XPath.XPathNavigator.Matches(System.Xml.XPath.XPathExpression)** method are specific to that method. A string representing an XPath expression.

#### Evaluate

[C#] public virtual object Evaluate(string xpath);

[C++] public: virtual Object\* Evaluate(String\* xpath);

[VB] Overridable Public Function Evaluate(ByVal xpath As String) As Object

[JScript] public function Evaluate(xpath : String) : Object;

#### *Description*

Evaluates the string representing an XPath expression and returns the typed result (int, boolean or string).

*Return Value:* The result of the expression.

The following C# code converts the Price/text() node to a number, multiplies it by 10, and returns the resulting value. A string representing an XPath expression that can be evaluated.

#### Evaluate

[C#] public virtual object Evaluate(XPathExpression expr);  
[C++] public: virtual Object\* Evaluate(XPathExpression\* expr);  
[VB] Overridable Public Function Evaluate(ByVal expr As XPathExpression) As Object  
[JScript] public function Evaluate(expr : XPathExpression) : Object; Evaluates the given expression and returns the typed result.

#### Description

Evaluates the **System.Xml.XPath.XPathExpression** and returns the typed result (int, boolean or string).

*Return Value:* The result of the expression.

The following C# code returns a number after converting the Price/text() node to a number and multiplying the value by 10. An **XPathExpression** that can be evaluated.

#### Evaluate

[C#] public virtual object Evaluate(XPathExpression expr, XPathNodeIterator context);  
[C++] public: virtual Object\* Evaluate(XPathExpression\* expr, XPathNodeIterator\* context);

```

1 [VB] Overridable Public Function Evaluate(ByVal expr As XPathExpression,
2   ByVal context As XPathNodeIterator) As Object
3 [JScript] public function Evaluate(expr : XPathExpression, context :
4   XPathNodeIterator) : Object;

```

### *Description*

Evaluates the **System.Xml.XPath.XPathExpression** using the supplied context and returns the typed result (int, boolean or string).

*Return Value:* The result of the expression.

The expression is evaluated using the **System.Xml.XPath.XPathNodeIterator.Current** node of the **XPathNodeIterator** as the context node. If *context* is **null**, the node on which the **XPathNavigator** is currently positioned is used as the context node. An **XPathExpression** that can be evaluated. An **System.Xml.XPath.XPathNodeIterator** pointing to the selected node set that the evaluation should be performed on.

### *GetAttribute*

```

19 [C#] public abstract string GetAttribute(string localName, string namespaceURI);
20 [C++] public: virtual String* GetAttribute(String* localName, String*
21   namespaceURI) = 0;
22 [VB] MustOverride Public Function GetAttribute(ByVal localName As String,
23   ByVal namespaceURI As String) As String
24 [JScript] public abstract function GetAttribute(localName : String, namespaceURI
25   : String) : String;

```

## Description

When overridden in a derived class, gets the value of the attribute with the specified **System.Xml.XPath.XPathNavigator.LocalName** and **System.Xml.XPath.XPathNavigator.NamespaceURI**.

*Return Value:* The value of the specified attribute or **String.Empty** if a matching attribute is not found.

The **XPathNavigator** should be positioned on an element node before calling this method. The local name of the attribute. The namespace URI of the attribute.

## GetNamespace

[C#] public abstract string GetNamespace(string name);

[C++] public: virtual String\* GetNamespace(String\* name) = 0;

[VB] MustOverride Public Function GetNamespace(ByVal name As String) As String

[JScript] public abstract function GetNamespace(name : String) : String;

## Description

When overridden in a derived class, returns the value of the namespace node corresponding to the specified local name.

*Return Value:* The value of the namespace node or **String.Empty** if a matching node is not found.

For a definition of namespace nodes, see section 5.4 of the W3C XML Path Language (XPath) recommendation located at <http://www.w3.org/TR/xpath#data-model> . The local name of the namespace node.

#### IsDescendant

[C#] public virtual bool IsDescendant(XPathNavigator nav);

[C++] public: virtual bool IsDescendant(XPathNavigator\* nav);

[VB] Overridable Public Function IsDescendant(ByVal nav As XPathNavigator)

As Boolean

[JScript] public function IsDescendant(nav : XPathNavigator) : Boolean;

#### *Description*

Determines whether the specified **XPathNavigator** is a descendant of the current **XPathNavigator** .

*Return Value:* **true** if *nav* is a descendant of the current navigator; otherwise **false**

.

A navigator is a descendant of another navigator if it is positioned on a descendant node of the current navigator. For example, using the following XML string: widget If the current navigator is positioned on the item node and *nav* is positioned on the name node, **IsDescendant** returns **true** . The **XPathNavigator** that you want to compare against.

#### IsSamePosition

[C#] public abstract bool IsSamePosition(XPathNavigator other);

[C++] public: virtual bool IsSamePosition(XPathNavigator\* other) = 0;



[VB] MustOverride Public Function IsSamePosition(ByVal other As  
XPathNavigator) As Boolean

[JScript] public abstract function IsSamePosition(other : XPathNavigator) :  
Boolean;

#### *Description*

When overridden in a derived class, determines whether the current  
**XPathNavigator** is at the same position as the specified **XPathNavigator** .

*Return Value:* **true** if *other* and this navigator have the same position; otherwise,  
**false** .

This method assumes that *other* is an **XPathNavigator** that shares the same  
implementation and is pointing at the same document instance as the current  
navigator. The **XPathNavigator** that you want to compare against.

#### *Matches*

[C#] public virtual bool Matches(string xpath);

[C++] public: virtual bool Matches(String\* xpath);

[VB] Overridable Public Function Matches(ByVal xpath As String) As Boolean

[JScript] public function Matches(xpath : String) : Boolean;

#### *Description*

Determines whether the current node matches the specified XPath  
expression.

*Return Value:* **true** if the current node matches the XPath expression; otherwise,  
**false** .

This method has no effect on the state of the **XPathNavigator** . An XPath expression.

### Matches

[C#] public virtual bool Matches(XPathExpression expr);

[C++] public: virtual bool Matches(XPathExpression\* expr);

[VB] Overridable Public Function Matches(ByVal expr As XPathExpression) As Boolean

[JScript] public function Matches(expr : XPathExpression) : Boolean; Determines whether the current node matches the specified expression.

### Description

Determines whether the current node matches the specified

**System.Xml.XPath.XPathExpression** .

*Return Value:* **true** if the current node matches the **XPathExpression** ; otherwise, **false** .

This method has no effect on the state of the **XPathNavigator** . An **XPathExpression** .

### MoveTo

[C#] public abstract bool MoveTo(XPathNavigator other);

[C++] public: virtual bool MoveTo(XPathNavigator\* other) = 0;

[VB] MustOverride Public Function MoveTo(ByVal other As XPathNavigator) As Boolean

[JScript] public abstract function MoveTo(other : XPathNavigator) : Boolean;

## Description

When overridden in a derived class, moves to the same position as the specified **XPathNavigator**.

*Return Value:* **true** if successful; otherwise **false**. If **false**, the current navigator is returned to its position before the call.

This method always returns **true** if *other* is an **XPathNavigator** that shares the same implementation and is pointing at the same document instance as the current navigator. The **XPathNavigator** positioned on the node that you want to move to.

## MoveToAttribute

```
[C#] public abstract bool MoveToAttribute(string localName, string  
namespaceURI);
```

```
[C++] public: virtual bool MoveToAttribute(String* localName, String*  
namespaceURI) = 0;
```

```
[VB] MustOverride Public Function MoveToAttribute(ByVal localName As  
String, ByVal namespaceURI As String) As Boolean
```

```
[JScript] public abstract function MoveToAttribute(localName : String,  
namespaceURI : String) : Boolean;
```

## Description

When overridden in a derived class, moves to the attribute with matching **System.Xml.XPath.XPathNavigator.LocalName** and **System.Xml.XPath.XPathNavigator.NamespaceURI**.

*Return Value:* **true** if the attribute is found; otherwise, **false** . If **false** , the position of the navigator does not change.

The navigator should be positioned on an element when calling this method. The local name of the attribute. The namespace URI of the attribute.

#### MoveToFirst

[C#] public abstract bool MoveToFirst();

[C++] public: virtual bool MoveToFirst() = 0;

[VB] MustOverride Public Function MoveToFirst() As Boolean

[JScript] public abstract function MoveToFirst() : Boolean;

#### *Description*

When overridden in a derived class, moves to the first sibling of the current node.

*Return Value:* **true** if there is a first sibling node; **false** if there is no first sibling or if the navigator is currently positioned on an attribute node.

#### MoveToFirstAttribute

[C#] public abstract bool MoveToFirstAttribute();

[C++] public: virtual bool MoveToFirstAttribute() = 0;

[VB] MustOverride Public Function MoveToFirstAttribute() As Boolean

[JScript] public abstract function MoveToFirstAttribute() : Boolean;

#### *Description*

When overridden in a derived class, moves to the first attribute.

*Return Value:* **true** if there is an attribute for the **XPathNavigator** to move to; otherwise, **false** .

The navigator should be positioned on an element when calling this method; otherwise, this method returns **false** and the position of the navigator does not change.

### MoveToFirstChild

[C#] public abstract bool MoveToFirstChild();

[C++] public: virtual bool MoveToFirstChild() = 0;

[VB] MustOverride Public Function MoveToFirstChild() As Boolean

[JScript] public abstract function MoveToFirstChild() : Boolean;

### Description

When overridden in a derived class, moves to the first child of the current node.

*Return Value:* **true** if there is a first child node; otherwise **false** .

**Root** and **Element** are the only two XPath node types that can have children. This property always returns **false** for all other node types.

### MoveToFirstNamespace

[C#] public abstract bool MoveToFirstNamespace();

[C++] public: virtual bool MoveToFirstNamespace() = 0;

[VB] MustOverride Public Function MoveToFirstNamespace() As Boolean

[JScript] public abstract function MoveToFirstNamespace() : Boolean;

## Description

When overridden in a derived class, moves the **XPathNavigator** to first namespace node of the current element.

*Return Value:* **true** if there is a first namespace node; otherwise **false** .

The navigator should be positioned on an element node when calling this method. After the navigator has been moved to the namespace node the **System.Xml.XPath.XPathNavigator.Name** property reflects the name of the namespace.

## MoveToId

[C#] public abstract bool MoveToId(string id);

[C++] public: virtual bool MoveToId(String\* id) = 0;

[VB] MustOverride Public Function MoveToId(ByVal id As String) As Boolean

[JScript] public abstract function MoveToId(id : String) : Boolean;

## Description

When overridden in a derived class, moves to the node that has an attribute of type ID whose value matches the specified string.

*Return Value:* **true** if the move was successful; otherwise **false** . If **false** , the navigator is returned to its position before the call.

This method can be used to identify nodes by unique ID provided the source document explicitly declares attributes of type ID using a DTD. The ID value of the node to which you want to move. This argument does not need to be atomized.

## MoveToNamespace

```
[C#] public abstract bool MoveToNamespace(string name);
[C++] public: virtual bool MoveToNamespace(String* name) = 0;
[VB] MustOverride Public Function MoveToNamespace(ByVal name As String)
As Boolean
[JScript] public abstract function MoveToNamespace(name : String) : Boolean;
```

### *Description*

When overridden in a derived class, moves the **XPathNavigator** to the namespace node with the specified local name.

**Return Value:** **true** if the move was successful; **false** if a matching namespace node was not found or if the navigator is not positioned on an element node.

The navigator should be positioned on an element node when calling this method. After the navigator has been moved to the namespace node the **System.Xml.XPath.XPathNavigator.Name** property reflects the name of the namespace. The local name of the namespace node.

## MoveToNext

```
[C#] public abstract bool MoveToNext();
[C++] public: virtual bool MoveToNext() = 0;
[VB] MustOverride Public Function MoveToNext() As Boolean
[JScript] public abstract function MoveToNext() : Boolean;
```

### *Description*

When overridden in a derived class, moves to the next sibling of the current node.

*Return Value:* **true** if there is a next sibling node; **false** if there are no more siblings or if the navigator is currently positioned on an attribute node.

#### MoveToNextAttribute

[C#] public abstract bool MoveToNextAttribute();

[C++] public: virtual bool MoveToNextAttribute() = 0;

[VB] MustOverride Public Function MoveToNextAttribute() As Boolean

[JScript] public abstract function MoveToNextAttribute() : Boolean;

#### *Description*

When overridden in a derived class, moves to the next attribute.

*Return Value:* **true** if there is a next attribute; **false** if there are no more attributes.

The navigator should be positioned on an element when calling this method; otherwise, this method returns **false** and the position of the navigator does not change.

#### MoveToNextNamespace

[C#] public abstract bool MoveToNextNamespace();

[C++] public: virtual bool MoveToNextNamespace() = 0;

[VB] MustOverride Public Function MoveToNextNamespace() As Boolean

[JScript] public abstract function MoveToNextNamespace() : Boolean;

#### *Description*



When overridden in a derived class, moves the **XPathNavigator** to the next namespace node.

*Return Value:* **true** if there is a next namespace node; otherwise **false** .

The navigator should be positioned on a namespace node when calling this method.

#### MoveToParent

[C#] public abstract bool MoveToParent();

[C++] public: virtual bool MoveToParent() = 0;

[VB] MustOverride Public Function MoveToParent() As Boolean

[JScript] public abstract function MoveToParent() : Boolean;

#### Description

When overridden in a derived class, moves to the parent of the current node.

*Return Value:* **true** if there is a parent node; otherwise **false** .

The following table lists the **System.Xml.XPath.XPathNodeType** and what type of parent node they may have, if any.

#### MoveToPrevious

[C#] public abstract bool MoveToPrevious();

[C++] public: virtual bool MoveToPrevious() = 0;

[VB] MustOverride Public Function MoveToPrevious() As Boolean

[JScript] public abstract function MoveToPrevious() : Boolean;

## Description

When overridden in a derived class, moves to the previous sibling of the current node.

**Return Value:** **true** if there is a previous sibling node; **false** if there is no previous sibling or if the navigator is currently positioned on an attribute node.

## MoveToRoot

[C#] public abstract void MoveToRoot();

[C++] public: virtual void MoveToRoot() = 0;

[VB] MustOverride Public Sub MoveToRoot()

[JScript] public abstract function MoveToRoot();

## Description

When overridden in a derived class, moves to the root node to which the current node belongs.

All nodes belong to one and only one document. Therefore, this method is always successful. For a definition of a root node, see section 5.1 of the W3C XML Path Language (XPath) recommendation located at <http://www.w3.org/TR/xpath#data-model>.

## Select

[C#] public virtual XPathNodeIterator Select(string xpath);

[C++] public: virtual XPathNodeIterator\* Select(String\* xpath);

[VB] Overridable Public Function Select(ByVal xpath As String) As

XPathNodeIterator

[JScript] public function Select(xpath : String) : XPathNodeIterator;

#### *Description*

Selects a node set using the specified XPath expression.

*Return Value:* An **System.Xml.XPath.XPathNodeIterator** pointing to the selected node set.

The context for the selection is the position of the navigator when you called this method. After calling this method, the **XPathNodeIterator** returned represents the set of selected nodes. Use **System.Xml.XPath.XPathNodeIterator.MoveNext** on the **XPathNodeIterator** to walk the selected node set. Use any of the **Move** methods on the **XPathNavigator** to move within the current node tree. The **XPathNavigator** navigation methods are independent of the selected nodes and the **XPathNodeIterator**. A string representing an XPath expression.

Select

[C#] public virtual XPathNodeIterator Select(XPathExpression expr);

[C++] public: virtual XPathNodeIterator\* Select(XPathExpression\* expr);

[VB] Overridable Public Function Select(ByVal expr As XPathExpression) As

XPathNodeIterator

[JScript] public function Select(expr : XPathExpression) : XPathNodeIterator;

Selects a node set using the specified XPath expression.

#### *Description*

Selects a node set using the specified

**System.Xml.XPath.XPathExpression** .

*Return Value:* An **System.Xml.XPath.XPathNodeIterator** pointing to the selected node set.

The context for the selection is the position of the navigator when you called this method. After calling this method, the **XPathNodeIterator** returned represents the set of selected nodes. Use **System.Xml.XPath.XPathNodeIterator.MoveNext** on the **XPathNodeIterator** to walk the selected node set. Use any of the **Move** methods on the **XPathNavigator** to move within the current node tree. The **XPathNavigator** navigation methods are independent of the selected nodes and the **XPathNodeIterator** . An **XPathExpression** .

SelectAncestors

[C#] public virtual XPathNodeIterator SelectAncestors(XPathNodeType type, bool matchSelf);

[C++] public: virtual XPathNodeIterator\* SelectAncestors(XPathNodeType type, bool matchSelf);

[VB] Overridable Public Function SelectAncestors(ByVal type As XPathNodeType, ByVal matchSelf As Boolean) As XPathNodeIterator

[JScript] public function SelectAncestors(type : XPathNodeType, matchSelf : Boolean) : XPathNodeIterator; Selects all the ancestor element nodes of the current node matching the selection criteria.

*Description*

1 Selects all the ancestor element nodes of the current node with the matching  
2 **System.Xml.XPath.XPathNodeType** .

3 *Return Value:* An **System.Xml.XPath.XPathNodeIterator** pointing to the  
4 selected nodes.

5 This method has no effect on the state of the **XPathNavigator** . The  
6 **XPathNodeType** of the ancestor nodes. **true** to include the context node in the  
7 selection; otherwise **false** .

#### 8 SelectAncestors

9  
10 [C#] public virtual XPathNodeIterator SelectAncestors(string name, string  
11 namespaceURI, bool matchSelf);

12 [C++] public: virtual XPathNodeIterator\* SelectAncestors(String\* name, String\*  
13 namespaceURI, bool matchSelf);

14 [VB] Overridable Public Function SelectAncestors(ByVal name As String, ByVal  
15 namespaceURI As String, ByVal matchSelf As Boolean) As XPathNodeIterator

16 [JScript] public function SelectAncestors(name : String, namespaceURI : String,  
17 matchSelf : Boolean) : XPathNodeIterator;

#### 18 19 *Description*

20 Selects all the ancestor element nodes of the current node with the supplied  
21 local name and namespace URI.

22 *Return Value:* An **System.Xml.XPath.XPathNodeIterator** pointing to the  
23 selected nodes.  
24  
25

This method has no effect on the state of the **XPathNavigator** . The local name of the ancestor nodes. The namespace URI of the ancestor nodes. **true** to include the context node in the selection; otherwise **false** .

#### SelectChildren

[C#] public virtual XPathNodeIterator SelectChildren(XPathNodeType type);  
 [C++] public: virtual XPathNodeIterator\* SelectChildren(XPathNodeType type);  
 [VB] Overridable Public Function SelectChildren(ByVal type As XPathNodeType) As XPathNodeIterator  
 [JScript] public function SelectChildren(type : XPathNodeType) : XPathNodeIterator; Selects all the child nodes of the current node matching the selection criteria.

#### *Description*

Selects all the child nodes of the current node with the matching **System.Xml.XPath.XPathNodeType** .

*Return Value:* An **System.Xml.XPath.XPathNodeIterator** pointing to the selected nodes.

This method has no effect on the state of the **XPathNavigator** . The **XPathNodeType** of the child nodes.

#### SelectChildren

[C#] public virtual XPathNodeIterator SelectChildren(string name, string namespaceURI);  
 [C++] public: virtual XPathNodeIterator\* SelectChildren(String\* name, String\*

namespaceURI);

[VB] Overridable Public Function SelectChildren(ByVal name As String, ByVal namespaceURI As String) As XPathNodeIterator

[JScript] public function SelectChildren(name : String, namespaceURI : String) : XPathNodeIterator;

### *Description*

Selects all the child element nodes of the current node with the supplied local name and namespace URI.

*Return Value:* An **System.Xml.XPath.XPathNodeIterator** pointing to the selected nodes.

This method has no effect on the state of the **XPathNavigator**. The local name of the child nodes. The namespace URI of the child nodes.

### **SelectDescendants**

[C#] public virtual XPathNodeIterator SelectDescendants(XPathNodeType type, bool matchSelf);

[C++] public: virtual XPathNodeIterator\* SelectDescendants(XPathNodeType type, bool matchSelf);

[VB] Overridable Public Function SelectDescendants(ByVal type As XPathNodeType, ByVal matchSelf As Boolean) As XPathNodeIterator

[JScript] public function SelectDescendants(type : XPathNodeType, matchSelf : Boolean) : XPathNodeIterator; Selects all the descendant nodes of the current node matching the selection criteria.

## *Description*

Selects all the descendant nodes of the current node with the matching

**System.Xml.XPath.XPathNodeType** .

*Return Value:* An **System.Xml.XPath.XPathNodeIterator** pointing to the selected nodes.

This method has no effect on the state of the **XPathNavigator** . The **XPathNodeType** of the descendant nodes. **true** to include the context node in the selection; otherwise **false** .

### SelectDescendants

[C#] public virtual XPathNodeIterator SelectDescendants(string name, string namespaceURI, bool matchSelf);

[C++] public: virtual XPathNodeIterator\* SelectDescendants(String\* name, String\* namespaceURI, bool matchSelf);

[VB] Overridable Public Function SelectDescendants(ByVal name As String, ByVal namespaceURI As String, ByVal matchSelf As Boolean) As

XPathNodeIterator

[JScript] public function SelectDescendants(name : String, namespaceURI : String, matchSelf : Boolean) : XPathNodeIterator;

## *Description*

Selects all the descendant element nodes of the current node with the supplied local name and namespace URI.



*Return Value:* An **System.Xml.XPath.XPathNodeIterator** pointing to the selected nodes.

This method has no effect on the state of the **XPathNavigator** . The local name of the descendant nodes. The namespace URI of the descendant nodes. **true** to include the context node in the selection; otherwise **false** .

**ICloneable.Clone**

[C#] object ICloneable.Clone();

[C++] Object\* ICloneable::Clone();

[VB] Function Clone() As Object Implements ICloneable.Clone

[JScript] function ICloneable.Clone() : Object;

**ToString**

[C#] public override string ToString();

[C++] public: String\* ToString();

[VB] Overrides Public Function ToString() As String

[JScript] public override function ToString() : String;

### *Description*

Gets the text value of the current node. This method is equivalent to returning the **System.Xml.XPath.XPathNavigator.Value** property.

*Return Value:* The content returned depends on the

**System.Xml.XPath.XPathNavigator.NodeType** of the node.

**XPathNodeIterator** class (System.Xml.XPath)

**ToString**

### Description

Provides an iterator over a set of selected nodes.

XPathNodeIterator

*Example Syntax:*

ToString

[C#] protected XPathNodeIterator();

[C++] protected: XPathNodeIterator();

[VB] Protected Sub New()

[JScript] protected function XPathNodeIterator();

Count

ToString

[C#] public virtual int Count {get;}

[C++] public: \_\_property virtual int get\_Count();

[VB] Overridable Public ReadOnly Property Count As Integer

[JScript] public function get Count() : int;

### Description

Gets the index of the last node in the selected set of nodes.

The **Count** property can be used to check whether nodes have been selected. The following C# code shows an example of this.

Current

ToString

```
[C#] public abstract XPathNavigator Current {get;}
[C++] public: __property virtual XPathNavigator* get_Current() = 0;
[VB] MustOverride Public ReadOnly Property Current As XPathNavigator
[JScript] public abstract function get Current() : XPathNavigator;
```

### Description

When overridden in a derived class, returns the navigator for this **XPathNodeIterator** positioned on the current node.

You can use the properties of the **XPathNavigator** to return information on the current node. However, the **XPathNavigator** cannot be used to move away from the selected node set. Doing so could invalidate the state of the navigator. Alternatively, you can clone the **XPathNavigator**. The cloned **XPathNavigator** can then be moved away from the selected node set. This is an application level decision. Providing this functionality may affect the performance of the XPath query.

CurrentPosition

ToString

```
[C#] public abstract int CurrentPosition {get;}
[C++] public: __property virtual int get_CurrentPosition() = 0;
[VB] MustOverride Public ReadOnly Property CurrentPosition As Integer
[JScript] public abstract function get CurrentPosition() : int;
```

## Description

When overridden in a derived class, gets the index of the current position in the selected set of nodes.

## Clone

[C#] public abstract XPathNodeIterator Clone();

[C++] public: virtual XPathNodeIterator\* Clone() = 0;

[VB] MustOverride Public Function Clone() As XPathNodeIterator

[JScript] public abstract function Clone() : XPathNodeIterator;

## Description

When overridden in a derived class, creates a new **XPathNodeIterator** .

**Return Value:** A new **XPathNodeIterator** object.

The cloned **XPathNodeIterator** is not affected by subsequent changes to the current **XPathNodeIterator** .

## MoveNext

[C#] public abstract bool MoveNext();

[C++] public: virtual bool MoveNext() = 0;

[VB] MustOverride Public Function MoveNext() As Boolean

[JScript] public abstract function MoveNext() : Boolean;

## Description

When overridden in a derived class, moves the **System.Xml.XPath.XPathNavigator** to the next node in the selected set.

*Return Value:* **true** if the **XPathNavigator** moved to the next node; **false** if there are no more selected nodes.

The node set is created in document order, therefore calling this method moves to the next node in document order. The **XPathNodeIterator** is positioned on the first node in the selected set after the initial call to **MoveNext** . This makes it easy to write a while loop.

**ICloneable.Clone**

[C#] object ICloneable.Clone();

[C++] Object\* ICloneable::Clone();

[VB] Function Clone() As Object Implements ICloneable.Clone

[JScript] function ICloneable.Clone() : Object;

XPathNodeType enumeration (System.Xml.XPath)

**ToString**

### *Description*

Specifies the XPath node types which can be returned from the **System.Xml.XPath.XPathNavigator** .

**ToString**

[C#] public const XPathNodeType All;

[C++] public: const XPathNodeType All;

[VB] Public Const All As XPathNodeType

[JScript] public var All : XPathNodeType;

*Description*

All node types.

ToString

[C#] public const XPathNodeType Attribute;

[C++] public: const XPathNodeType Attribute;

[VB] Public Const Attribute As XPathNodeType

[JScript] public var Attribute : XPathNodeType;

*Description*

An attribute in the node tree.

ToString

[C#] public const XPathNodeType Comment;

[C++] public: const XPathNodeType Comment;

[VB] Public Const Comment As XPathNodeType

[JScript] public var Comment : XPathNodeType;

*Description*

A comment.

ToString

1  
2 [C#] public const XPathNodeType Element;  
3 [C++] public: const XPathNodeType Element;  
4 [VB] Public Const Element As XPathNodeType  
5 [JScript] public var Element : XPathNodeType;  
6

7 *Description*

8 An element in the node tree.

9 ToString  
10

11 [C#] public const XPathNodeType Namespace;  
12 [C++] public: const XPathNodeType Namespace;  
13 [VB] Public Const Namespace As XPathNodeType  
14 [JScript] public var Namespace : XPathNodeType;  
15

16 *Description*

17 A namespace node. This is equivalent to a DOM attribute that declares a  
18 namespace (for example, xmlns).

19 ToString  
20

21 [C#] public const XPathNodeType ProcessingInstruction;  
22 [C++] public: const XPathNodeType ProcessingInstruction;  
23 [VB] Public Const ProcessingInstruction As XPathNodeType  
24 [JScript] public var ProcessingInstruction : XPathNodeType;  
25

*Description*

A processing instruction.

ToString

[C#] public const XPathNodeType Root;

[C++] public: const XPathNodeType Root;

[VB] Public Const Root As XPathNodeType

[JScript] public var Root : XPathNodeType;

*Description*

The root of the node tree.

ToString

[C#] public const XPathNodeType SignificantWhitespace;

[C++] public: const XPathNodeType SignificantWhitespace;

[VB] Public Const SignificantWhitespace As XPathNodeType

[JScript] public var SignificantWhitespace : XPathNodeType;

*Description*

A node with whitespace and xml:space set to preserve.

ToString

[C#] public const XPathNodeType Text;

[C++] public: const XPathNodeType Text;



1 [VB] Public Const Text As XPathNodeType

2 [JScript] public var Text : XPathNodeType;

3  
4 *Description*

5 The text content of an element or attribute.

6 ToString

7  
8 [C#] public const XPathNodeType Whitespace;

9 [C++] public: const XPathNodeType Whitespace;

10 [VB] Public Const Whitespace As XPathNodeType

11 [JScript] public var Whitespace : XPathNodeType;

12  
13 *Description*

14 A node with only whitespace characters and no significant whitespace.

15 XPathResultType enumeration (System.Xml.XPath)

16 ToString

17  
18  
19 *Description*

20 Specifies the the return type of the XPath expression.

21 ToString

22  
23 [C#] public const XPathResultType Any;

24 [C++] public: const XPathResultType Any;

25 [VB] Public Const Any As XPathResultType

[JScript] public var Any : XPathResultType;

*Description*

Any of the XPath node types.

ToString

[C#] public const XPathResultType Boolean;

[C++] public: const XPathResultType Boolean;

[VB] Public Const Boolean As XPathResultType

[JScript] public var Boolean : XPathResultType;

*Description*

Boolean value **true** or **false** .

ToString

[C#] public const XPathResultType Error;

[C++] public: const XPathResultType Error;

[VB] Public Const Error As XPathResultType

[JScript] public var Error : XPathResultType;

*Description*

The expression does not evaluate to the correct XPath type.

ToString

[C#] public const XPathResultType Navigator;

[C++] public: const XPathResultType Navigator;  
 [VB] Public Const Navigator As XPathResultType  
 [JScript] public var Navigator : XPathResultType;

*Description*

A tree fragment.

ToString

[C#] public const XPathResultType NodeSet;  
 [C++] public: const XPathResultType NodeSet;  
 [VB] Public Const NodeSet As XPathResultType  
 [JScript] public var NodeSet : XPathResultType;

*Description*

A node collection.

ToString

[C#] public const XPathResultType Number;  
 [C++] public: const XPathResultType Number;  
 [VB] Public Const Number As XPathResultType  
 [JScript] public var Number : XPathResultType;

*Description*

Numeric value.

ToString

[C#] public const XPathResultType String;

[C++] public: const XPathResultType String

### **System.Xml.Xsl**

This namespace provides support for XSL/T transformations.

Type Name	Description
System.Xml.Xsl.IXsltContextFunction	Provides an interface to a given function defined in the XSLT stylesheet during runtime execution.
System.Xml.Xsl.IXsltContextVariable	Provides an interface to a given variable that is defined in the stylesheet during runtime execution.
System.Xml.Xsl.XsltArgumentList	Contains a variable number of arguments which are either XSLT parameters or extension objects.
System.Xml.Xsl.XsltCompileException	The exception that is thrown by the <b>System.Xml.Xsl.XsltTransform.Load(System.String)</b> method when an error is found in the XSLT stylesheet. <b>XsltCompileException</b>
System.Xml.Xsl.XsltContext	Encapsulates the current execution context of the XSLT processor allowing XPath to resolve functions, parameters, and

	namespaces within XPath expressions. <b>abstract</b>
System.Xml.Xsl.XsltException	The exception that is thrown when an error occurs while processing an XSL transform. <b>XsltException</b>
System.Xml.Xsl.XsltTransform	Transforms XML data using an XSLT stylesheet.

### *Description*

This namespace provides support for XSL/T transformations.

IXsltContextFunction interface (System.Xml.Xsl)

### *Description*

Provides an interface to a given function defined in the XSLT stylesheet during runtime execution.

Properties:

ArgTypes

[C#] XPathResultType[] ArgTypes {get;}

[C++] XPathResultType get\_ArgTypes();

[VB] ReadOnly Property ArgTypes As XPathResultType ()

[JScript] abstract function get ArgTypes() : XPathResultType[];

## *Description*

Gets the supplied XPath types for the function's argument list. This information can be used to discover the function's signature which allows you to differentiate between overloaded functions.

### *Maxargs*

[C#] int Maxargs {get;}

[C++] int get\_Maxargs();

[VB] ReadOnly Property Maxargs As Integer

[JScript] abstract function get Maxargs() : int;

## *Description*

Gets the maximum number of arguments for the function. This enables the user to differentiate between overloaded functions.

### *Minargs*

[C#] int Minargs {get;}

[C++] int get\_Minargs();

[VB] ReadOnly Property Minargs As Integer

[JScript] abstract function get Minargs() : int;

## *Description*

Gets the minimum number of arguments for the function. This enables the user to differentiate between overloaded functions.

## ReturnType

[C#] XPathResultType ReturnType {get;}

[C++] XPathResultType get\_ReturnType();

[VB] ReadOnly Property ReturnType As XPathResultType

[JScript] abstract function get ReturnType() : XPathResultType;

### Description

Gets the **System.Xml.XPath.XPathResultType** representing the XPath type returned by the function.

### Methods:

#### Invoke

[C#] object Invoke(XsltContext xsltContext, object[] args, XPathNavigator docContext);

[C++] Object\* Invoke(XsltContext\* xsltContext, Object\* args \_\_gc[], XPathNavigator\* docContext);

[VB] Function Invoke(ByVal xsltContext As XsltContext, ByVal args() As Object, ByVal docContext As XPathNavigator) As Object

[JScript] function Invoke(xsltContext : XsltContext, args : Object[], docContext : XPathNavigator) : Object;

### Description

Provides the method to Invoke the function with the given arguments in the given namespace.

*Return Value:* An **System.Object** representing the return value of the function.

The *docContext* parameter is required so that functions which require a node-set can be supplied the current XSLT context. For example, consider the following XSLT document function:

#### EXEMPLARY COMPUTING SYSTEM AND ENVIRONMENT

Fig. 4 illustrates an example of a suitable computing environment 400 within which the programming framework 132 may be implemented (either fully or partially). The computing environment 400 may be utilized in the computer and network architectures described herein.

The exemplary computing environment 400 is only one example of a computing environment and is not intended to suggest any limitation as to the scope of use or functionality of the computer and network architectures. Neither should the computing environment 400 be interpreted as having any dependency or requirement relating to any one or combination of components illustrated in the exemplary computing environment 400.

The framework 132 may be implemented with numerous other general purpose or special purpose computing system environments or configurations. Examples of well known computing systems, environments, and/or configurations that may be suitable for use include, but are not limited to, personal computers, server computers, multiprocessor systems, microprocessor-based systems, network PCs, minicomputers, mainframe computers, distributed computing environments



1 that include any of the above systems or devices, and so on. Compact or subset  
2 versions of the framework may also be implemented in clients of limited  
3 resources, such as cellular phones, personal digital assistants, handheld computers,  
4 or other communication/computing devices.

5 The framework 132 may be described in the general context of computer-  
6 executable instructions, such as program modules, being executed by one or more  
7 computers or other devices. Generally, program modules include routines,  
8 programs, objects, components, data structures, etc. that perform particular tasks  
9 or implement particular abstract data types. The framework 132 may also be  
10 practiced in distributed computing environments where tasks are performed by  
11 remote processing devices that are linked through a communications network. In  
12 a distributed computing environment, program modules may be located in both  
13 local and remote computer storage media including memory storage devices.

14 The computing environment 400 includes a general-purpose computing  
15 device in the form of a computer 402. The components of computer 402 can  
16 include, by are not limited to, one or more processors or processing units 404, a  
17 system memory 406, and a system bus 408 that couples various system  
18 components including the processor 404 to the system memory 406.

19 The system bus 408 represents one or more of several possible types of bus  
20 structures, including a memory bus or memory controller, a peripheral bus, an  
21 accelerated graphics port, and a processor or local bus using any of a variety of  
22 bus architectures. By way of example, such architectures can include an Industry  
23 Standard Architecture (ISA) bus, a Micro Channel Architecture (MCA) bus, an  
24 Enhanced ISA (EISA) bus, a Video Electronics Standards Association (VESA)

1 local bus, and a Peripheral Component Interconnects (PCI) bus also known as a  
2 Mezzanine bus.

3 Computer 402 typically includes a variety of computer readable media.  
4 Such media can be any available media that is accessible by computer 402 and  
5 includes both volatile and non-volatile media, removable and non-removable  
6 media.

7 The system memory 406 includes computer readable media in the form of  
8 volatile memory, such as random access memory (RAM) 410, and/or non-volatile  
9 memory, such as read only memory (ROM) 412. A basic input/output system  
10 (BIOS) 414, containing the basic routines that help to transfer information  
11 between elements within computer 402, such as during start-up, is stored in ROM  
12 412. RAM 410 typically contains data and/or program modules that are  
13 immediately accessible to and/or presently operated on by the processing unit 404.

14 Computer 402 may also include other removable/non-removable,  
15 volatile/non-volatile computer storage media. By way of example, Fig. 4  
16 illustrates a hard disk drive 416 for reading from and writing to a non-removable,  
17 non-volatile magnetic media (not shown), a magnetic disk drive 418 for reading  
18 from and writing to a removable, non-volatile magnetic disk 420 (e.g., a “floppy  
19 disk”), and an optical disk drive 422 for reading from and/or writing to a  
20 removable, non-volatile optical disk 424 such as a CD-ROM, DVD-ROM, or other  
21 optical media. The hard disk drive 416, magnetic disk drive 418, and optical disk  
22 drive 422 are each connected to the system bus 408 by one or more data media  
23 interfaces 426. Alternatively, the hard disk drive 416, magnetic disk drive 418,  
24 and optical disk drive 422 can be connected to the system bus 408 by one or more  
25 interfaces (not shown).

The disk drives and their associated computer-readable media provide non-volatile storage of computer readable instructions, data structures, program modules, and other data for computer 402. Although the example illustrates a hard disk 416, a removable magnetic disk 420, and a removable optical disk 424, it is to be appreciated that other types of computer readable media which can store data that is accessible by a computer, such as magnetic cassettes or other magnetic storage devices, flash memory cards, CD-ROM, digital versatile disks (DVD) or other optical storage, random access memories (RAM), read only memories (ROM), electrically erasable programmable read-only memory (EEPROM), and the like, can also be utilized to implement the exemplary computing system and environment.

Any number of program modules can be stored on the hard disk 416, magnetic disk 420, optical disk 424, ROM 412, and/or RAM 410, including by way of example, an operating system 426, one or more application programs 428, other program modules 430, and program data 432. Each of the operating system 426, one or more application programs 428, other program modules 430, and program data 432 (or some combination thereof) may include elements of the programming framework 132.

A user can enter commands and information into computer 402 via input devices such as a keyboard 434 and a pointing device 436 (e.g., a "mouse"). Other input devices 438 (not shown specifically) may include a microphone, joystick, game pad, satellite dish, serial port, scanner, and/or the like. These and other input devices are connected to the processing unit 404 via input/output interfaces 440 that are coupled to the system bus 408, but may be connected by

1 other interface and bus structures, such as a parallel port, game port, or a universal  
2 serial bus (USB).

3 A monitor 442 or other type of display device can also be connected to the  
4 system bus 408 via an interface, such as a video adapter 444. In addition to the  
5 monitor 442, other output peripheral devices can include components such as  
6 speakers (not shown) and a printer 446 which can be connected to computer 402  
7 via the input/output interfaces 440.

8 Computer 402 can operate in a networked environment using logical  
9 connections to one or more remote computers, such as a remote computing device  
10 448. By way of example, the remote computing device 448 can be a personal  
11 computer, portable computer, a server, a router, a network computer, a peer device  
12 or other common network node, and so on. The remote computing device 448 is  
13 illustrated as a portable computer that can include many or all of the elements and  
14 features described herein relative to computer 402.

15 Logical connections between computer 402 and the remote computer 448  
16 are depicted as a local area network (LAN) 450 and a general wide area network  
17 (WAN) 452. Such networking environments are commonplace in offices,  
18 enterprise-wide computer networks, intranets, and the Internet.

19 When implemented in a LAN networking environment, the computer 402 is  
20 connected to a local network 450 via a network interface or adapter 454. When  
21 implemented in a WAN networking environment, the computer 402 typically  
22 includes a modem 456 or other means for establishing communications over the  
23 wide network 452. The modem 456, which can be internal or external to computer  
24 402, can be connected to the system bus 408 via the input/output interfaces 440 or  
25 other appropriate mechanisms. It is to be appreciated that the illustrated network

1 connections are exemplary and that other means of establishing communication  
2 link(s) between the computers 402 and 448 can be employed.

3 In a networked environment, such as that illustrated with computing  
4 environment 400, program modules depicted relative to the computer 402, or  
5 portions thereof, may be stored in a remote memory storage device. By way of  
6 example, remote application programs 458 reside on a memory device of remote  
7 computer 448. For purposes of illustration, application programs and other  
8 executable program components such as the operating system are illustrated herein  
9 as discrete blocks, although it is recognized that such programs and components  
10 reside at various times in different storage components of the computing device  
11 402, and are executed by the data processor(s) of the computer.

12 An implementation of the framework 132, and particularly, the API 142 or  
13 calls made to the API 142, may be stored on or transmitted across some form of  
14 computer readable media. Computer readable media can be any available media  
15 that can be accessed by a computer. By way of example, and not limitation,  
16 computer readable media may comprise "computer storage media" and  
17 "communications media." "Computer storage media" include volatile and non-  
18 volatile, removable and non-removable media implemented in any method or  
19 technology for storage of information such as computer readable instructions, data  
20 structures, program modules, or other data. Computer storage media includes, but  
21 is not limited to, RAM, ROM, EEPROM, flash memory or other memory  
22 technology, CD-ROM, digital versatile disks (DVD) or other optical storage,  
23 magnetic cassettes, magnetic tape, magnetic disk storage or other magnetic storage  
24 devices, or any other medium which can be used to store the desired information  
25 and which can be accessed by a computer.

1           “Communication media” typically embodies computer readable  
2 instructions, data structures, program modules, or other data in a modulated data  
3 signal, such as carrier wave or other transport mechanism. Communication media  
4 also includes any information delivery media. The term “modulated data signal”  
5 means a signal that has one or more of its characteristics set or changed in such a  
6 manner as to encode information in the signal. By way of example, and not  
7 limitation, communication media includes wired media such as a wired network or  
8 direct-wired connection, and wireless media such as acoustic, RF, infrared, and  
9 other wireless media. Combinations of any of the above are also included within  
10 the scope of computer readable media.

11           Alternatively, portions of the framework may be implemented in hardware  
12 or a combination of hardware, software, and/or firmware. For example, one or  
13 more application specific integrated circuits (ASICs) or programmable logic  
14 devices (PLDs) could be designed or programmed to implement one or more  
15 portions of the framework.

## 17           Conclusion

18           Although the invention has been described in language specific to structural  
19 features and/or methodological acts, it is to be understood that the invention  
20 defined in the appended claims is not necessarily limited to the specific features or  
21 acts described. Rather, the specific features and acts are disclosed as exemplary  
22 forms of implementing the claimed invention.